# Secure Interval Skyline Queries over Encrypted Time Series Data

By

Songnian Zhang, Suprio Ray, Rongxing Lu, Yandong Zheng, Yunguo Guan and Jun Shao

Technique Report TR21-242
July 20, 2021

Faculty of Computer Science
University of New Brunswick
Fredericton, N.B. E3B 5A3
Canada

Phone: (506) 453-4566
Fax: (506) 453-3566
E-mail: fcs@unb.ca
http://www.cs.unb.ca

# Secure Interval Skyline Queries over Encrypted Time Series Data

Songnian Zhang†, Suprio Ray†, Rongxing Lu†, Yandong Zheng†, Yunguo Guan†, Jun Shao‡
† *University of New Brunswick, Fredericton, Canada.* Email:{szhang17, sray, rlu1, yzheng8, yguan4}@unb.ca
‡ *Zhejiang Gongshang University, Hangzhou, 310018, China.* Email: chn.junshao@gmail.com

***Abstract*—As interval skyline queries can find time series with dominating advantages, it is practically useful in real-world applications for supporting analytics over time series data. Recently, outsourcing the encrypted time series data has been widely adopted by the data owner for economic considerations. However, it inevitably lowers data utility and query efficiency. Existing secure skyline query schemes either leak critical information or are inefficient. In this paper, we propose an efficient and privacy-preserving interval skyline query scheme based on symmetric homomorphic encryption (SHE) to cope with the above issues. Specifically, we first design two secure subprotocols for basic operations: big than comparison and equality test, and the propose two privacy-preserving logic gates to achieve quick permutation and hide access patterns. Using the above subprotocols and gates, we devise secure sort and secure dominance check protocols to securely sort time series data and determine dominance relations, respectively. With these protocols, we finally propose our secure skyline computation protocol that can ensure both security and efficiency. To deal with the characteristics of time series data, we design a secure high-dimensional dominance check protocol to improve performance and a look-up table to index time series for quick query response. Detailed security analysis shows that our proposed scheme is indeed privacy-preserving. With extensive experiments, we evaluate our proposed scheme and compare the core components of our scheme with the state-of-the-art solution. The results show that our protocols outperform the compared solution by two orders of magnitude in the computational cost and at least 23× in the communication cost.**

## I. Introduction

TIME series data occurs ubiquitously across human endeavors and has a wide range of applications, including medical and biological experimental observations, social activity mining, electricity consumption monitoring, and so forth [1]–[3]. Consequently, analyzing time series data has attracted extensive investigations [1]–[7]. Among them, interval skyline over time series data, i.e., segments of time series that show dominating advantages over others, is of particular interest [5]–[7] since it can capture the time series that have highest time series value in a query interval. A real-world example is illustrated as follows.

*Example 1 (Motivation). A hospital provides online medical monitoring for patients, and a patient's heart rate can be captured by a time series. Fig. 1 shows four time series (four patients) with the value of beats per minute (bpm) in a period from $t_1$ to $t_8$. Assume a doctor would like to analyze the heart rate of different patients by asking "which patients have high bpm in the time interval from $t_2$ to $t_7$?". $\mathtt{s}_1$ and $\mathtt{s}_3$ are interesting to the doctor. That is because $\mathtt{s}_1$ has the highest average bpm, while $\mathtt{s}_3$ has the highest bps at $t_5$. Regarding $\mathtt{s}_2$ and $\mathtt{s}_4$, they are ignored since $\mathtt{s}_1$ is higher than both of them at each timestamp of the query interval $t_2$-$t_7$.*

| | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $t_5$ | $t_6$ | $t_7$ | $t_8$ |
|---|---|---|---|---|---|---|---|---|
| $\mathtt{s}_1$ | 81 | 80 | 84 | 82 | 79 | 83 | 80 | 81 |
| $\mathtt{s}_2$ | 76 | 79 | 80 | 77 | 76 | 80 | 75 | 77 |
| $\mathtt{s}_3$ | 74 | 76 | 75 | 80 | 85 | 81 | 79 | 82 |
| $\mathtt{s}_4$ | 75 | 75 | 77 | 74 | 75 | 75 | 76 | 74 |

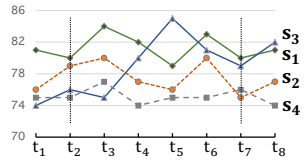



Fig. 1. An example of the interval skyline query over time series data.

In the practical example, finding interesting time series $\{\mathtt{s}_1, \mathtt{s}_3\}$ can be achieved by performing an interval skyline query. Technically, a time series $\mathtt{s}$ is returned if, in the query interval, there does not exist another time series $\mathtt{s}'$ that is not worse than $\mathtt{s}$ at each timestamp and is better than $\mathtt{s}$ at least one timestamp [5]. See the formal definition of interval skyline query in Section III-A.

One of the characteristics of the time series data is that it involves continuous updates over time, which indicates the service providers should keep online at all times to receive the reported data from entities. With the data volume growing, the service providers may pay a great price to stay online and maintain the data. Accordingly, they tend to outsource their data and the corresponding services, i.e., interval skyline query in this paper, to a third-party cloud for reaping economic benefits. However, it inevitably raises privacy issues since the real-world time series data, e.g., the medical data in the above example, may often contain sensitive information. A promising solution is to encrypt outsourced data and perform queries over encrypted time series data.

However, existing privacy-preserving skyline schemes investigate the encrypted multi-dimensional data rather than time series data [8]–[11]. Although their core component, i.e., securely determining dominance relation (whether one data record dominates another one), can be used in the privacy-preserving interval skyline scheme, they are either leaking critical information or inefficient. For example, the solution in [11] leaks the order relations of each dimension, while the solution in [9] is expensive though it is secure (see detailed analysis and other solutions in Section VIII). Therefore, in this paper, our goal is to propose a privacy-preserving interval skyline query scheme while ensuring efficiency.

**Challenges**. i) Privacy preservation. In addition to the security of time series data, our proposed scheme should preserve single-dimensional privacy and hide access patterns. That is

because revealing these information may incur inference attacks [12], [13]. Here, single-dimensional privacy indicates the order or equality relation of values in each dimension (timestamp). However, in skyline computation, it is essential for an operator to determine the dominance relation by obtaining the order relation of each dimension. Therefore, it is challenging to compute skyline without leaking single-dimensional privacy. For the access pattern, it is required for the operator to select skyline points without knowing which ones are selected. As a result, ensuring these privacy is challenging in performing interval skyline queries. ii) Efficiency. To determine dominance relation without leaking privacy, the state-of-the-art solution [9] employs two cloud servers to cooperatively determine the order relation of dimensions one by one, which is expensive in the communication overhead. Therefore, it raises the question "is there a more efficient solution to determine dominance relation without compromising privacy?"; iii) Time series data. The characteristics of time series data elicit new challenges for computing interval skyline. First, time series data usually involves many timestamps. It indicates that we need to process high-dimensional data if we treat each timestamp as a dimension, which deteriorates the performance of the existing solutions. Second, time series data usually continues updates over time. It is not easy to dynamically index these time series data as little storage cost as possible to quickly respond to interval skyline queries.

Aiming at the above challenges, in this paper, we propose a novel interval skyline computation scheme over encrypted time series data, in which a lightweight symmetric homomorphic encryption (SHE) scheme is adopted as the cryptographic primitive. Our proposed scheme can preserve the privacy of plaintexts, single-dimensional privacy, and access patterns while ensuring efficiency. Specifically, the main contributions of this paper are four-fold as follows.

- First, we propose Secure Bigger Than (SBT) and Secure Equal (SEQ) subprotocols to securely and efficiently determine the bigger than and equal to relations, respectively. Besides, we observe a good property of XOR and XNOR gates, i.e., one can quickly obtain the encrypted output of the gates without leaking it if one input is a plaintext, and the other is a ciphertext. Based on the observation, we propose privacy-preserving XOR and XNOR gates. The former can be used in quick permutation, while the latter can be used to hide access patterns. All of them will serve as the building blocks of our proposed scheme.
- Second, we design a Secure Sort (SS) and Secure Dominance Check (SDC) protocols to securely sort time series data according to their sum values and determine dominance relations, respectively. In the SS protocol, the privacy-preserving XOR gate is used to achieve a lightweight position permutation. As the core component in skyline computation, our SDC protocol adopts an efficient approach to determine dominance relations without checking order relations for all dimensions one by one. Meanwhile, a flip-coin mechanism is adopted in the SDC protocol to ensure single-dimensional privacy. Further, we carefully devise a Secure High-dimensional Dominance Check (SHDC) protocol to deal with the high-dimensional time series data, in which a dominance check tree, denoted as DC-tree, is presented to improve performance.
- Third, based on the SS, SDC, and SHDC protocols, we propose a secure skyline computation protocol to compute skyline over high-dimensional time series data, in which we employ a new algorithm by modifying the sort-filter-skyline (SFS) algorithm [14]. In addition, to quickly respond to the interval skyline query, we design a two-dimensional look-up table to index the time series data. It balances the storage costs and computational costs to cope with continuous updates of time series data.
- Finally, we conduct extensive experiments to evaluate the performance of our proposed scheme and compare the core component: SDC and SHDC protocols, with the state-of-the-art solution [9]. The results show that our protocols are much more efficient than the compared solution two orders of magnitude in the computational cost and at least 23× in the communication cost.

This paper is organized as follows. In Section II, we introduce our system model and security model. Then, we review our preliminaries in Section III. After that, we introduce the building blocks in Section IV and present our proposed scheme in Section V, followed by security analysis and performance evaluation in Section VI and Section VII, respectively. Finally, we discuss some related works in Section VIII and draw our conclusion in Section IX.

## II. Models and Design Goal

In this section, we formalize our system and security models.

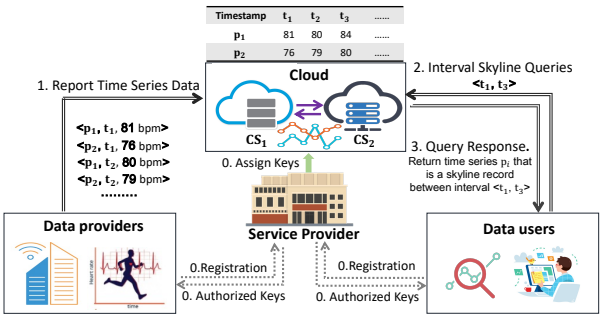



Fig. 2. System model under consideration

### A. System Model

In our system model, we consider a typical cloud-based interval skyline query model, which mainly consists of four types of entities: a service provider $\mathcal{SP}$, a set of data providers $\mathcal{P} = \{p_1, p_2, \cdots\}$, a cloud $\mathcal{C}$ with two servers $\{\mathcal{CS}_1, \mathcal{CS}_2\}$, and multiple data users $\mathcal{U}$=$\{u_1, u_2, \cdots\}$, as shown in Fig. 2.

Service Provider $\mathcal{SP}$: In our system model, $\mathcal{SP}$ is an initiator for the whole system. On the one hand, $\mathcal{SP}$ offers registration services to other entities. On the other hand, $\mathcal{SP}$ is responsible for providing interval skyline query services to registered data users. However, since $\mathcal{SP}$ has limited storage and computing resources, it outsources the time series data to a cloud and employs the cloud to offer the interval skyline query

services to users. Before offering these services, $\mathcal{SP}$ generates secret keys and securely distributes them to different entities.

Data Providers $\mathcal{P}$=$\{p_1, p_2, \cdots\}$: We consider a registered smart device as a data provider. A data provider $p_i$ can collect data and report it to the cloud at a certain time interval. The format of the reported data is: $\langle$id, timestamp, value$\rangle$. Taking a patient with the chronic heart disease as an example, the implanted sensor of the patient reports the heart rate, i.e., beats per minute (bpm), to the cloud as the format $\langle p_1, t_i, \text{bpm}\rangle$, where $p_1$ is id of the sensor, and $t_i$ $(i = 1, 2, \cdots)$ denotes the timestamp. For the reported values, we assume all of them are integers. It is reasonable since we can easily convert non-integer data into integers [15].

Cloud $\mathcal{C}$: In our system, $\mathcal{SP}$ employs two cloud servers $\mathcal{C} = \{\mathcal{CS}_1, \mathcal{CS}_2\}$ from different cloud server providers. Both of them are considered as powerful in storage and computing resources. They will manage the reported time series data and cooperatively offer reliable interval skyline query services to data users.

Data users $\mathcal{U}$=$\{u_1, u_2, ...\}$: In the system, only the authorized data users can enjoy the interval skyline query services from the cloud $\mathcal{C}$. That is, in order to obtain desired results from $\mathcal{C}$, data users must register to $\mathcal{SP}$ before launching the interval skyline query requests.

### B. Security Model

In our security model, since $\mathcal{SP}$ is the service organizer and has no motivation to deviate from it, he/she is considered as *fully trusted*. That is, $\mathcal{SP}$ will honestly generate and distribute secret keys and sincerely provide registration services to other entities. For data providers $\mathcal{P}$ and data users $\mathcal{U}$, we consider the authorized ones are *honest*, i.e., they will honestly report time series data and launch the interval skyline queries, respectively. However, in our model, the cloud services $\mathcal{CS}_1$ and $\mathcal{CS}_2$ are considered as *honest-but-curious*. They will faithfully follow the designed protocols and schemes but may be curious to learn the private information. For example, the cloud servers may be interested in the reported heart rate to determine whether the owner of the sensor has heart disease. As a result, to ensure privacy, the data providers report the encrypted time series data: $\langle$id, timestamp, encrypted value$\rangle$. Nonetheless, the cloud still attempts to obtain the private information, including the plaintexts of the encrypted values and query results, in the process of interval skyline queries. Note that we assume there is no collusion between $\mathcal{CS}_1$ and $\mathcal{CS}_2$, as well as no collusion between the cloud and other entities. It is reasonable since the cloud should maintain its reputation and interests. Note that, since we focus on the privacy preservation, the external attacks, e.g., Denial of Service (DoS) attacks, are beyond the scope of this paper and will be discussed in our future work.

## III. Preliminaries

In this section, we first formally define the interval skyline query. Then, we introduce the symmetric homomorphic encryption (SHE) scheme, which is the cryptographic primitive employed in our proposed scheme.

### A. Interval Skyline Query

The interval skyline query is used to compute skyline on time series data, which was first introduced in [5]. Before delving into the details, we first present the time series data.

A time series $\mathtt{s}$ is composed of a sequence of pairs $\langle t_i, \text{value}\rangle$ ordered by $t_i$, where $t_i$ $(i = 1, 2, \cdots)$ are timestamps. We denote the value of $\mathtt{s}$ at timestamp $t_i$ as $\mathtt{s}[t_i]$. Following the assumption in [5], all time series are synchronized, i.e., each time series $\mathtt{s}$ holds a value $\mathtt{s}[t_i]$ on a timestamp $t_i > 0$.

**Definition 1** (Time Interval)**.** *A time interval $[t_i : t_j]$ indicates a range in time that contains the set of timestamps existing between $t_i$ and $t_j$ $(t_i < t_j)$. We say that $\mathtt{s}[t_i, t_j] = (\mathtt{s}[t_i], \mathtt{s}[t_{i+1}], \cdots, \mathtt{s}[t_j])$ is a subsequence of time series $\mathtt{s}$ in the time interval $[t_i : t_j]$.*

With the above definition, here we formally define the interval skyline query as follows.

**Definition 2** (Interval dominance)**.** *Given two time series $\mathtt{s}_1$ and $\mathtt{s}_2$, $\mathtt{s}_1$ is said to dominate $\mathtt{s}_2$ in a time interval $[t_i : t_j]$, denoted as $\mathtt{s}_1 \succ_{[t_i:t_j]} \mathtt{s}_2$, if $\forall t_k \in [t_i, t_j], \mathtt{s}_1[t_k] \geq \mathtt{s}_2[t_k]$ and $\exists t_l \in [t_i, t_j], \mathtt{s}_1[t_l] > \mathtt{s}_2[t_l]$.*

**Definition 3** (Interval Skyline Query)**.** *Given a set $\mathcal{S}$ with $n$ time series and a time interval $[t_i : t_j]$, the interval skyline query returns a set $\mathcal{S}_{\text{sky}} \subseteq \mathcal{S}$, in which the time series are not dominated by any other time series in $\mathcal{S}$. That is, $\mathcal{S}_{\text{sky}} = \{\mathtt{s}_k \in \mathcal{S} \mid \nexists \mathtt{s}_l \in \mathcal{S}$ such that $\mathtt{s}_l \succ_{[t_i:t_j]} \mathtt{s}_k\}$. Note that each time series in $\mathcal{S}_{\text{sky}}$ only contains values between $t_i$ and $t_j$.*

### B. Symmetric Homomorphic Encryption

SHE is an efficient symmetric homomorphic encryption scheme that can support homomorphic addition and multiplication. It was first proposed in [16] and then proved to be IND-CPA secure in [17]. Concretely, SHE includes three algorithms, namely i) key generation $\mathsf{KeyGen}()$; ii) encryption $\mathsf{Enc}()$; and iii) decryption $\mathsf{Dec}()$, as follows:

- $\mathsf{KeyGen}(k_0, k_1, k_2)$: Given three security parameters $\{k_0, k_1, k_2\}$ satisfying $k_1 \ll k_2 < k_0$, the algorithm first chooses two large prime numbers $p, q$ with $|p| = |q| = k_0$ and sets $\mathcal{N} = pq$. Then, it generates the secret key $sk = (p, \mathcal{L})$, where $\mathcal{L}$ is a random number with $|\mathcal{L}| = k_2$, and the public parameter $pp = (k_0, k_1, k_2, \mathcal{N})$. Besides, the algorithm sets the basic message space $\mathcal{M} = [-2^{k_1-1}, 2^{k_1-1})$.
- $\mathsf{Enc}(sk, m)$: On input of a secret key $sk$ and a message $m \in \mathcal{M}$, the encryption algorithm outputs the ciphertext $\mathsf{E}(m) = (r\mathcal{L} + m)(1 + r'p) \mod \mathcal{N}$, where $r \in \{0,1\}^{k_2}$ and $r' \in \{0,1\}^{k_0}$ are random numbers.
- $\mathsf{Dec}(sk, \mathsf{E}(m))$: Taking the secret key $sk$ and a ciphertext $\mathsf{E}(m)$ as inputs, the algorithm recovers a message $m' = (\mathsf{E}(m) \mod p) \mod \mathcal{L} = (r\mathcal{L} + m) \mod \mathcal{L}$. If $m' < \frac{\mathcal{L}}{2}$, it indicates $m \geq 0$ and $m = m'$. Otherwise, $m < 0$ and $m = m' - \mathcal{L}$.

SHE satisfies the homomorphic addition and multiplication properties as follows: i) Homomorphic addition-I: $\mathsf{E}(m_1) + \mathsf{E}(m_2) \mod \mathcal{N} \rightarrow \mathsf{E}(m_1 + m_2)$; ii) Homomorphic multiplication-I: $\mathsf{E}(m_1) \cdot \mathsf{E}(m_2) \mod \mathcal{N} \rightarrow \mathsf{E}(m_1 \cdot m_2)$;

iii) Homomorphic addition-II: $\mathsf{E}(m_1) + m_2 \mod \mathcal{N} \rightarrow \mathsf{E}(m_1 + m_2)$; iv) Homomorphic multiplication-II: $\mathsf{E}(m_1) \cdot m_2 \mod \mathcal{N} \rightarrow \mathsf{E}(m_1 \cdot m_2)$ when $m_2 > 0$.

**Encryption with public key**. In order to realize the SHE encryption under public key setting, we take $sk = (p, \mathcal{L})$ as the private key and use it to generate two ciphertexts $\mathsf{E}(0)_1$, $\mathsf{E}(0)_2$ of 0 with different random numbers. Then, the public key is set as $pk = \{\mathsf{E}(0)_1, \mathsf{E}(0)_2, pp\}$. In such a way, one can use the above homomorphic properties to encrypt a message $m$ by the following

$$\mathsf{E}(m) = m + r_1 \cdot \mathsf{E}(0)_1 + r_2 \cdot \mathsf{E}(0)_2 \mod \mathcal{N} \quad (1)$$

where $r_1$ and $r_2 \in \{0,1\}^{k_2}$ are two random numbers. This approach has been proven to be IND-CPA secure in [18].

SHE is a leveled fully homomorphic encryption scheme, since it has limited number of operation of Homomorphic multiplication-I. To tackle it, we adopt a bootstrapping protocol [17] to support an infinite number of Homomorphic multiplication-I. We refer the readers to [17] for more details.

## IV. Building Blocks

In this section, we introduce our designed secure subprotocols and privacy-preserving logic gates, which serve as the building blocks for constructing more complex protocols.

### A. Secure Subprotocols

To achieve our privacy-preserving interval skyline query scheme, we need to compute some basic functions on encrypted data. Here, we design Secure Bigger Than (SBT) and Secure Equal (SEQ) subprotocols to determine whether two given encrypted inputs (messages) have the bigger than and equal to relations, respectively. Both of them are deployed in a two-server model, where $\mathcal{CS}_1$ holds encrypted inputs and $pk$, and $\mathcal{CS}_2$ has the secret key $sk$. Our goal is to leak nothing to the cloud $\{\mathcal{CS}_1, \mathcal{CS}_2\}$ while ensuring efficiency of these subprotocols.

*1) Secure Bigger Than (SBT) Subprotocol:* Given two encrypted messages $\mathsf{E}(m_1)$ and $\mathsf{E}(m_2)$, the SBT subprotocol is to determine whether $m_1 > m_2$ without leaking any $m_1, m_2$ related information to $\mathcal{CS}_1$ or $\mathcal{CS}_2$. If $m_1 > m_2$, the subprotocol outputs $\mathsf{E}(1)$, otherwise $\mathsf{E}(0)$.

• Step-1: $\mathcal{CS}_1$ flips a coin $s \in \{-1, 1\}$ and chooses two random numbers $r_1, r_2 \in \{0,1\}^{k_1}$ satisfying $r_1 > r_2 > 0$. Then, $\mathcal{CS}_1$ computes:

$$\begin{aligned}\mathsf{E}(\theta) =& \mathsf{E}(s \cdot r_1) \cdot (\mathsf{E}(m_1) + \mathsf{E}(m_2) \cdot \mathsf{E}(-1)) + \mathsf{E}(s \cdot r_2) \cdot \mathsf{E}(-1) \\ =& \mathsf{E}(s \cdot r_1 \cdot (m_1 - m_2) - s \cdot r_2).\end{aligned}$$

After that, $\mathcal{CS}_1$ sends $\mathsf{E}(\theta)$ to $\mathcal{CS}_2$.

• Step-2: On receiving $\mathsf{E}(\theta)$, $\mathcal{CS}_2$ uses $sk$ to recover $\theta$ and checks whether $\theta > 0$. If yes, $\mathcal{CS}_2$ makes $\mu = 1$ and encrypts it into $\mathsf{E}(\mu)$. Otherwise, $\mathcal{CS}_2$ generates $\mathsf{E}(\mu) = \mathsf{E}(0)$. Next, $\mathcal{CS}_2$ returns $\mathsf{E}(\mu)$ to $\mathcal{CS}_1$.

• Step-3: If $s = 1$, $\mathcal{CS}_1$ lets $\mathsf{E}(\delta) = \mathsf{E}(\mu)$. If $s = -1$, $\mathcal{CS}_1$ computes $\mathsf{E}(\delta) = \mathsf{E}(1) + \mathsf{E}(\mu) \cdot \mathsf{E}(-1) = \mathsf{E}(1 - \mu)$. Finally, $\mathsf{E}(\delta)$ is the output of our SBT subprotocol.

**Correctness**. If $s = 1$, we have $\mathsf{E}(\theta) = \mathsf{E}(r_1 \cdot (m_1 - m_2) - r_2)$. When $m_1 > m_2$, $\theta > 0$. As a result, $\mathsf{E}(\delta) = \mathsf{E}(\mu) = \mathsf{E}(1)$. When $m_1 \leq m_2$, $\theta < 0$. In this case, $\mathsf{E}(\delta) = \mathsf{E}(\mu) = \mathsf{E}(0)$. Therefore, when $s = 1$, *iff* $m_1 > m_2$, the SBT subprotocol outputs $\mathsf{E}(\delta) = \mathsf{E}(1)$. On the one hand, if $s = -1$, we have $\mathsf{E}(\theta) = \mathsf{E}(r_1 \cdot (m_2 - m_1) + r_2)$. When $m_1 > m_2$, $\theta < 0$. As a result, $\mathsf{E}(\delta) = \mathsf{E}(1 - \mu) = \mathsf{E}(1 - 0) = \mathsf{E}(1)$. When $m_1 \leq m_2$, $\theta > 0$. In this case, $\mathsf{E}(\delta) = \mathsf{E}(1 - \mu) = \mathsf{E}(1 - 1) = \mathsf{E}(0)$. Therefore, when $s = -1$, *iff* $m_1 > m_2$, the SBT subprotocol outputs $\mathsf{E}(\delta) = \mathsf{E}(1)$. Thus, our SBT subprotocol is correct.

*2) Secure Equal (SEQ) Subprotocol:* Given two encrypted messages $\mathsf{E}(m_1)$ and $\mathsf{E}(m_2)$, the subprotocol is to determine whether $m_1 = m_2$ without leaking any $m_1, m_2$ related information to $\mathcal{CS}_1$ or $\mathcal{CS}_2$. If $m_1 = m_2$, the subprotocol outputs $\mathsf{E}(1)$, otherwise $\mathsf{E}(0)$.

• Step-1: $\mathcal{CS}_1$ flips a coin $s \in \{-1, 1\}$ and chooses two random numbers $r_1, r_2 \in \{0,1\}^{k_1}$ satisfying $r_1 > r_2 > 0$. Then, $\mathcal{CS}_1$ computes:

$$\begin{aligned}\mathsf{E}(\theta) =& \mathsf{E}(s \cdot r_1) \cdot \mathsf{E}((m_1 - m_2)^2) + \mathsf{E}(s \cdot r_2) \cdot \mathsf{E}(-1) \\ =& \mathsf{E}(s \cdot r_1 \cdot (m_1 - m_2)^2 - s \cdot r_2).\end{aligned}$$

After that, $\mathcal{CS}_1$ sends $\mathsf{E}(\theta)$ to $\mathcal{CS}_2$.

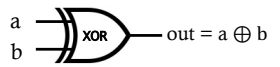
• Step-2: On receiving $\mathsf{E}(\theta)$, $\mathcal{CS}_2$ uses $sk$ to recover $\theta$ and checks whether $\theta < 0$. If yes, $\mathcal{CS}_2$ makes $\mu = 1$ and encrypts it into $\mathsf{E}(\mu)$. Otherwise, $\mathcal{CS}_2$ generates $\mathsf{E}(\mu) = \mathsf{E}(0)$. Next, $\mathcal{CS}_2$ returns $\mathsf{E}(\mu)$ to $\mathcal{CS}_1$.

• Step-3: If $s = 1$, $\mathcal{CS}_1$ lets $\mathsf{E}(\delta) = \mathsf{E}(\mu)$. If $s = -1$, $\mathcal{CS}_1$ computes $\mathsf{E}(\delta) = \mathsf{E}(1) + \mathsf{E}(\mu) \cdot \mathsf{E}(-1) = \mathsf{E}(1 - \mu)$. Finally, $\mathsf{E}(\delta)$ is the output of our SEQ subprotocol.

**Correctness**. If $s = 1$, we have $\mathsf{E}(\theta) = \mathsf{E}(r_1 \cdot (m_1 - m_2)^2 - r_2)$. When $m_1 = m_2$, $\theta < 0$. As a result, $\mathsf{E}(\delta) = \mathsf{E}(\mu) = \mathsf{E}(1)$. When $m_1 \neq m_2$, $\theta > 0$. In this case, $\mathsf{E}(\delta) = \mathsf{E}(\mu) = \mathsf{E}(0)$. Therefore, when $s = 1$, *iff* $m_1 = m_2$, the SEQ subprotocol outputs $\mathsf{E}(\delta) = \mathsf{E}(1)$. On the other hand, if $s = -1$, we have $\mathsf{E}(\theta) = \mathsf{E}(-r_1 \cdot (m_1 - m_2)^2 + r_2)$. When $m_1 = m_2$, $\theta > 0$. As a result, $\mathsf{E}(\delta) = \mathsf{E}(1 - \mu) = \mathsf{E}(1 - 0) = \mathsf{E}(1)$. When $m_1 \neq m_2$, $\theta < 0$. In this case, $\mathsf{E}(\delta) = \mathsf{E}(1 - \mu) = \mathsf{E}(1 - 1) = \mathsf{E}(0)$. Therefore, when $s = -1$, *iff* $m_1 = m_2$, the SEQ subprotocol outputs $\mathsf{E}(\delta) = \mathsf{E}(1)$. Thus, our SEQ subprotocol is correct.

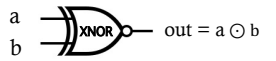
### B. Privacy-Preserving Logic Gates

In addition to the secure subprotocols, we also need some digital logic gates as building blocks to construct our proposed scheme. Here, we introduce two simple privacy-preserving logic gates: XOR and XNOR, to securely produce an encrypted output.




| Input | | Output |
|---|---|---|
| a | b | out |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

(a) XOR gate




| Input | | Output |
|---|---|---|
| a | b | out |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

(b) XNOR gate

Fig. 3. Gate symbol, boolean expression, and truth table of logic gates.

*1) Privacy-Preserving XOR Gate:* The XOR gate works by receiving two inputs, each designated with either 1 or 0. It outputs 0 if the two inputs are the same. Otherwise, the gate produces 1. Fig. 3(a) shows the gate symbol, boolean expression, and truth table of XOR gate, in which we observed: i) if $a = 0, out = b$; ii) if $a = 1, out = 1 - b$. If the input $a$ is in plaintext, and $b$ is in ciphertext, one can obtain the privacy-preserving XOR gate by using the observation as follows.

$$\mathsf{E}(out) = a \oplus \mathsf{E}(b) = \begin{cases} \mathsf{E}(b) & \text{if } a = 0 \\ \mathsf{E}(1-b) & \text{if } a = 1, \end{cases} \tag{2}$$

where $\mathsf{E}(1-b) = \mathsf{E}(1) + \mathsf{E}(b) \cdot \mathsf{E}(-1)$. The privacy-preserving XOR gate has the same truth table as the original gate, but can securely and efficiently compute the encrypted output $\mathsf{E}(out)$. Although the input $a$ is in plaintext, the output $\mathsf{E}(out)$ is kept secret, since the input $b$ is encrypted.

*2) Privacy-Preserving XNOR Gate:* The XNOR gate is the opposite of the XOR gate, i.e., it outputs 1 if the two inputs are the same. We demonstrate the gate symbol, boolean expression, and truth table of XNOR gate in Fig. 3(b). With an opposite observation of the XOR gate, our privacy-preserving XNOR gate can obtain the encrypted output as follows.

$$\mathsf{E}(out) = a \odot \mathsf{E}(b) = \begin{cases} \mathsf{E}(1-b) & \text{if } a = 0 \\ \mathsf{E}(b) & \text{if } a = 1, \end{cases} \tag{3}$$

Similarly, $\mathsf{E}(out)$ can be quickly computed from $\mathsf{E}(b)$ and is kept secret due to the encrypted input $\mathsf{E}(b)$.

## V. Our Proposed Scheme

In this section, based on the above building blocks, we first propose three novel secure protocols, which are key components in computing skylines. Then, we introduce two secure skyline computation protocols, in which one is the basic solution, and the other is the secure solution. Finally, we present our privacy-preserving interval skyline query scheme.

### A. Secure Protocols

Due to the high efficiency of the sort-based skyline computation algorithm, e.g., Sort Filter Skyline (SFS) [14], we design our efficient and privacy-preserving skyline computation protocols based on such an algorithm. However, as we know, it is challenging to protect the privacy of plaintexts and access patterns if we adopt the sort-based algorithm. It is because we have to sort the given dataset and check the dominance relations over the sorted dataset, which leaks the order relations and access patterns. To tackle it, we devise secure sort (SS) and secure dominance check (SDC) protocols to make the sort-based algorithm available while ensuring both efficiency and security. Furthermore, if we treat a timestamp as a dimension, the time series data usually has high-dimensional. To improve the efficiency, we design a secure high-dimensional dominance check (SHDC) protocol to specially determine the dominance relation for high-dimensional data.

*1) Secure Sort (SS) Protocol:* Assume $\mathcal{CS}_1$ has a set of $d$-dimensional data $\{\mathsf{E}(\vec{\mathrm{x}}_i) = (\mathsf{E}(\mathrm{x}_i^1), \mathsf{E}(\mathrm{x}_i^2), \cdots, \mathsf{E}(\mathrm{x}_i^d)) \mid \mathrm{x}_i^j \in \mathcal{M}, i \in [1,n], j \in [1,d]\}$ and their sum values $\mathsf{E}(\sigma_i) = \mathsf{E}(\sum_{j=1}^{d} \mathrm{x}_i^j)$, and $\mathcal{CS}_2$ holds $sk$. We also suppose that $\mathcal{CS}_1$ has randomly assigned a unique binary sequence $\{b_i = b_i^1 b_i^2 \cdots b_i^\rho \cdots b_i^{\lceil \log_2 n \rceil} \mid b_i^\rho \in \{0,1\}, \rho \in [1, \lceil \log_2 n \rceil]\}$ to each data record, as shown in Fig. 4. The goal of the SS protocol is to sort the encrypted dataset in descending order according to the sum values $\{\sigma_i \mid i \in [1,n]\}$ without leaking underlying plaintexts and access patterns to the cloud. That is, neither $\mathcal{CS}_1$ nor $\mathcal{CS}_2$ knows the plaintexts $\{(\mathrm{x}_i^1, \mathrm{x}_i^2, \cdots, \mathrm{x}_i^d, \sum_{j=1}^{d} \mathrm{x}_i^j) \mid i \in [1,n]\}$ or the link between the sorted dataset and the original dataset. We depict our SS protocol as follows.

- Step-1. First, $\mathcal{CS}_1$ generates $\lceil \log_2 n \rceil$ random bits $\{r^\rho \mid \rho \in [1, \lceil \log_2 n \rceil], r^\rho \in \{0,1\}\}$. After that, $\mathcal{CS}_1$ computes a new bit sequence for each data record, denoted as $\hat{b}_i^\rho$, by performing the XOR gate $\hat{b}_i^\rho = b_i^\rho \oplus r^\rho$ (not privacy-preserving XOR gate). Next, $\mathcal{CS}_1$ chooses $n+1$ random numbers $\{r_0, r_1, \cdots, r_n\}$ satisfying $r_0 > r_i, i \in [1,n]$. Finally, $\mathcal{CS}_1$ constructs a pair $\langle \mathsf{E}(\hat{\sigma}_i), \hat{b}_i \rangle$ for each data record and sends $n$ pairs to $\mathcal{CS}_2$, where $\mathsf{E}(\hat{\sigma}_i) = \mathsf{E}(r_0 \cdot \sigma_i + r_i)$ and $\hat{b}_i = \hat{b}_i^1 \hat{b}_i^2 \cdots \hat{b}_i^\rho \cdots \hat{b}_i^{\lceil \log_2 n \rceil}$.
- Step-2: On receiving these pairs $\{\langle \mathsf{E}(\hat{\sigma}_i), \hat{b}_i \rangle \mid i \in [1,n]\}$, $\mathcal{CS}_2$ first uses $sk$ to recover $\hat{\sigma}_i$ and sorts the bit sequences $\{\hat{b}_i \mid i \in [1,n]\}$ in a descending order according to the corresponding $\hat{\sigma}_i$. After sorting, $\mathcal{CS}_2$ encrypts each bit using SHE scheme: $\{\mathsf{E}(\hat{b}_i^\rho) \mid \rho \in [1, \lceil \log_2 n \rceil]\}$. Next, $\mathcal{CS}_2$ returns the sorted and encrypted bit sequences $\{\mathsf{E}(\hat{b}_i^\rho) \mid \rho \in [1, \lceil \log_2 n \rceil], i \in [1,n]\}$ to $\mathcal{CS}_1$.
- Step-3: With the random bits $\{r^\rho \mid \rho \in [1, \lceil \log_2 n \rceil], r^\rho \in \{0,1\}\}$ and the received bit sequences, $\mathcal{CS}_1$ adopts the privacy-preserving XOR gate, i.e., Eq. (2), to compute a set of new bit sequences $\{\mathsf{E}(\delta_i^\rho) = r^\rho \oplus \mathsf{E}(\hat{b}_i^\rho) \mid \rho \in [1, \lceil \log_2 n \rceil], i \in [1,n]\}$, which indicates the sorted index of original dataset and is the output of our SS protocol.

Note that we can further use the privacy-preserving XNOR gate to compute the sorted dataset. For the $k$-th data record, i.e., whose sum value is the $k$-th largest, one can obtain $\{\mathsf{E}(\mathrm{x}_k^j) \mid j \in [1,d]\}$ as follows.

$$\mathsf{E}(\mathrm{x}_k^j) = \sum\nolimits_{i=1}^{n} \mathsf{E}(\mathrm{x}_i^j) \cdot \prod\nolimits_{\rho=1}^{\lceil \log_2 n \rceil} (b_i^\rho \odot \mathsf{E}(\delta_k^\rho)) \tag{4}$$

Fig. 4 demonstrates an example of our secure sort protocol, in which there are four time series from Fig. 1. It is essential for $\mathcal{CS}_1$ to send bit sequences $\hat{b}_i$ to $\mathcal{CS}_2$ in our SS protocol, which guarantees the correctness when the number of data records $n < 2^{\lceil \log_2 n \rceil}$.

**Correctness**. We say our SS protocol is correct if the encrypted bit sequence set $\{\mathsf{E}(\delta_i^\rho) \mid \rho \in [1, \lceil \log_2 n \rceil], i \in [1,n]\}$ represents the sorted indexes $\{b_i^\rho \mid \rho \in [1, \lceil \log_2 n \rceil], i \in [1,n]\}$ according to the corresponding sum value $\sigma_i$. In our SS protocol, $\mathcal{CS}_2$ sorts the bit sequences $\{\hat{b}_i^\rho \mid \rho \in [1, \lceil \log_2 n \rceil], i \in [1,n]\}$ according to their sum values $\{\hat{\sigma}_i \mid i \in [1,n]\}$. Since $r_0 > r_i$ and $\hat{\sigma}_i = r_0 \cdot \sigma_i + r_i$, $\{\hat{\sigma}_i \mid i \in [1,n]\}$ keeps the order of $\{\sigma_i \mid i \in [1,n]\}$. Meanwhile, since the privacy-preserving XOR gate has the same truth table as the original XOR gate and $\hat{b}_i^\rho = b_i^\rho \oplus r^\rho$, we have $\delta_i^\rho = r^\rho \oplus \hat{b}_i^\rho = r^\rho \oplus b_i^\rho \oplus r^\rho = b_i^\rho$. Besides, the sorted $\{\hat{b}_i^\rho \mid \rho \in [1, \lceil \log_2 n \rceil], i \in [1,n]\}$ means the set $\{\mathsf{E}(\delta_i^\rho) \mid \rho \in [1, \lceil \log_2 n \rceil], i \in [1,n]\}$ is sorted. Thus,

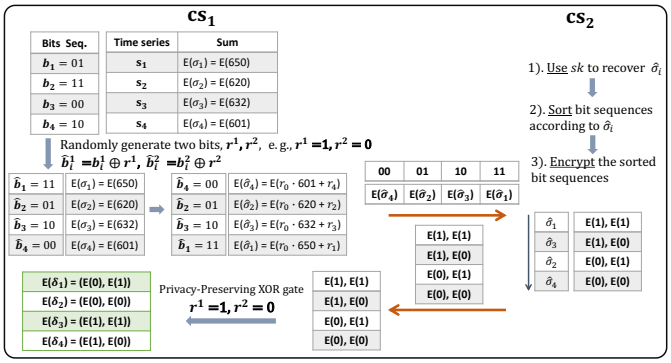



Fig. 4. Our Secure Sort Protocol.

$\{b_i^\rho \mid \rho \in [1, \lceil \log_2 n \rceil], i \in [1, n]\}$ is sorted according to the corresponding sum value $\sigma_i$.

*2) Secure Dominance Check (SDC) Protocol:* Assume $\mathcal{CS}_1$ has two encrypted $d$-dimensional data records $\mathsf{E}(\vec{\mathrm{x}}_1), \mathsf{E}(\vec{\mathrm{x}}_2)$ and $pk$ of SHE, while $\mathcal{CS}_2$ holds $sk$. The SDC protocol is to determine whether $\vec{\mathrm{x}}_1$ dominates $\vec{\mathrm{x}}_2$ without leaking the underlying plaintexts and the single-dimensional privacy to the cloud. If $\vec{\mathrm{x}}_1$ dominates $\vec{\mathrm{x}}_2$, our SDC protocol outputs $\mathsf{E}(1)$, otherwise $\mathsf{E}(0)$. Note that, since the $d$-dimensional data record can be treated as extracting a $d$ length time interval from the time series data, here the dominance relation between $\vec{\mathrm{x}}_1$ and $\vec{\mathrm{x}}_2$ is equivalent to the interval dominance (Definition 2) with any $d$ length time interval. We show the details of our SDC protocol as follows.

• Step-1. First, for the $i$th-dimension, $\mathcal{CS}_1$ flips a coin $s_i \in \{-1, 1\}$ and chooses two random numbers $r_1^i, r_2^i \in \{0,1\}^{k_1}$ satisfying $r_1^i > r_2^i > 0$. Then, $\mathcal{CS}_1$ computes $\mathsf{E}(\theta^i) = \mathsf{E}(s_i \cdot r_1^i \cdot (\mathrm{x}_1^i - \mathrm{x}_2^i) + s_i \cdot r_2^i)$. If $s_i = 1$, let $\alpha^i = 2^i$, otherwise $\alpha^i = 0$. Next, $\mathcal{CS}_1$ computes $\alpha = \sum_{i=1}^{d} \alpha^i$ and sends $\{\mathsf{E}(\theta^i) \mid i \in [1, d]\}$ to $\mathcal{CS}_2$.

• Step-2. On receiving $\{\mathsf{E}(\theta^i) \mid i \in [1, d]\}$, $\mathcal{CS}_2$ first uses $sk$ to recover $\theta^i$. If $\theta^i > 0$, $\mathcal{CS}_2$ computes $\beta^i = 2^i$. Otherwise $\beta^i = 0$. Afterward, $\mathcal{CS}_2$ computes $\beta = \sum_{i=1}^{d} \beta^i$ and encrypts it into $\mathsf{E}(\beta)$. Next, $\mathcal{CS}_2$ sends $\mathsf{E}(\beta)$ to $\mathcal{CS}_1$.

• Step-3. With $pk$, $\mathcal{CS}_1$ encrypts $\alpha$ into $\mathsf{E}(\alpha)$. Then, $\mathcal{CS}_1$ runs the SEQ subprotocol, where the inputs are $\mathsf{E}(\alpha)$ and $\mathsf{E}(\beta)$, to test whether $\alpha = \beta$. If yes, the SEQ subprotocol returns $\mathsf{E}(\delta_1) = \mathsf{E}(1)$, otherwise $\mathsf{E}(\delta_1) = \mathsf{E}(0)$.

• Step-4: $\mathcal{CS}_2$ first adds up all dimensions of $\mathsf{E}(\vec{\mathrm{x}}_1)$ and $\mathsf{E}(\vec{\mathrm{x}}_2)$, i.e., $\mathsf{E}(\sigma_1) = \mathsf{E}(\sum_{i=1}^{d} \mathrm{x}_1^i)$ and $\mathsf{E}(\sigma_2) = \mathsf{E}(\sum_{i=1}^{d} \mathrm{x}_2^i)$. Next, $\mathcal{CS}_1$ runs the SBT subprotocol to determine whether $\sigma_1 > \sigma_2$. If yes, the SBT subprotocol returns $\mathsf{E}(\delta_2) = \mathsf{E}(1)$, otherwise $\mathsf{E}(\delta_2) = \mathsf{E}(0)$.

• Step-5: With $\mathsf{E}(\delta_1)$ and $\mathsf{E}(\delta_2)$, $\mathcal{CS}_1$ computes $\mathsf{E}(\delta) = \mathsf{E}(\delta_1) \cdot \mathsf{E}(\delta_2) = \mathsf{E}(\delta_1 \cdot \delta_2)$ as the output of our SDC protocol.

**Correctness**. If $s_i = 1$, we have $\alpha^i = 2^i$. When $\mathrm{x}_1^i \geq \mathrm{x}_2^i$, $\theta^i = r_1^i \cdot (\mathrm{x}_1^i - \mathrm{x}_2^i) + r_2^i > 0$, leading to $\beta^i = 2^i$. As a result, $\alpha^i = \beta^i$. When $\mathrm{x}_1^i < \mathrm{x}_2^i$, it means $\theta^i < 0$, leading to $\beta^i = 0$. In this case, $\alpha^i \neq \beta^i$. Thus, when $s_i = 1$, *iff* $\forall i \in [1, d]$: $\mathrm{x}_1^i - \mathrm{x}_2^i \geq 0$, we have $\alpha = \beta$ and $\mathsf{E}(\delta_1) = \mathsf{E}(1)$. Similarly, we can prove that, when $s_i = -1$, *iff* $\forall i \in [1, d]$: $\mathrm{x}_1^i \geq \mathrm{x}_2^i$, we have $\mathsf{E}(\delta_1) = \mathsf{E}(1)$. Further, under the condition of $\forall i, \mathrm{x}_1^i \geq \mathrm{x}_2^i$, $\exists i, \mathrm{x}_1^i > \mathrm{x}_2^i$ indicates that $\mathtt{s}_1 > \mathtt{s}_2$. In this case, the STB subprotocol returns $\mathsf{E}(\delta_2) = \mathsf{E}(1)$. Thus, $\mathsf{E}(\delta) = \mathsf{E}(1 \cdot 1) = \mathsf{E}(1)$, *iff* $\vec{\mathrm{x}}_1$ dominates $\vec{\mathrm{x}}_2$.

*3) Secure High-dimensional Dominance Check (SHDC) Protocol:* With the same assumption as that of the SDC protocol, the SHDC protocol is to deal with the high-dimensional data, i.e., $d$ is large. The stricter assumption of the SHDC protocol is that the sum value of $\mathsf{E}(\vec{\mathrm{x}}_1)$ is no less than that of $\mathsf{E}(\vec{\mathrm{x}}_2)$, namely, $\mathsf{E}(\sigma_1) \geq \mathsf{E}(\sigma_2)$. In fact, it is easy to answer the assumption by applying our SS protocol. The main idea of our SHDC protocol is to convert the high-dimensional data to low dimensions and then apply our SDC protocol. To achieve it, we design a dominance check tree, denoted as DC-tree, in our SHDC protocol, which improves the performance when checking the dominance relation for high-dimensional data.

Here, we first introduce the basic information of the DC-tree. In our DC-tree, each non-leaf node has two children: left child and right child, and contains three fields: $\langle [j_1, j_2], \mathsf{E}(\sigma_1^{[j_1,j_2]}), \mathsf{E}(\sigma_2^{[j_1,j_2]}) \rangle$. For the leaf node, it also has three fields $\langle [j_1, j_2], \mathsf{E}(\vec{\mathrm{x}}_1^{[j_1,j_2]}), \mathsf{E}(\vec{\mathrm{x}}_2^{[j_1,j_2]}) \rangle$. We list the detailed description of these fields in Table I.

TABLE I
FIELDS OF LEAF AND NON-LEAF NODES

| Field | Description |
|---|---|
| $[j_1, j_2]$ | a dimension range, $[j_1, j_2] \subseteq [1, d]$ |
| $\mathsf{E}(\sigma_i^{[j_1,j_2]})$ $(i = 1, 2)$ | the encrypted sum value of $\mathsf{E}(\vec{\mathrm{x}}_i)$ in the dimension range $[j_1, j_2]$. $\mathsf{E}(\sigma_i^{[j_1,j_2]}) = \sum_{j=j_1}^{j_2} \mathsf{E}(\mathrm{x}_i^j)$ |
| $\mathsf{E}(\vec{\mathrm{x}}_i^{[j_1,j_2]})$ $(i = 1, 2)$ | the sub vector of $\mathsf{E}(\vec{\mathrm{x}}_i)$ extracting from $j_1$ to $j_2$ dimension. |

Unlike the traditional tree-based data structure used to retrieve data, our DC-tree is used to navigate and facilitate the dominance check. Specifically, the process of tree building is the running of our SHDC protocol. When the building process stops, our SHDC protocol outputs the dominance relation between $\mathsf{E}(\vec{\mathrm{x}}_1)$ and $\mathsf{E}(\vec{\mathrm{x}}_2)$. Next, we depict the process of our SHDC protocol as follows.

• Step-1. $\mathcal{CS}_1$ first uses the permutation $\pi$ to permute these two data records $(\mathsf{E}(\mathrm{x}_i^1), \mathsf{E}(\mathrm{x}_i^2), \cdots, \mathsf{E}(\mathrm{x}_i^d))$ as $(\mathsf{E}(\mathrm{x}_i^{\pi(1)}), \mathsf{E}(\mathrm{x}_i^{\pi(2)}), \cdots, \mathsf{E}(\mathrm{x}_i^{\pi(d)}))$, where $i = 1, 2$. To simplify the description, we ignore the permutation symbol $\pi$ in the protocol. Then, $\mathcal{CS}_1$ constructs the root node. Since $\mathsf{E}(\sigma_1^{[1,d]})$ must be no less than $\mathsf{E}(\sigma_2^{[1,d]})$, we mark the root node as $\langle [1, d], \perp, \perp \rangle$. After that, $\mathcal{CS}_1$ divides the $d$ dimensional data records $\mathsf{E}(\vec{\mathrm{x}}_1)$ and $\mathsf{E}(\vec{\mathrm{x}}_2)$ into two parts, respectively, in which the left part is $\mathsf{E}(\vec{\mathrm{x}}_i^l) = (\mathsf{E}(\mathrm{x}_i^1), \mathsf{E}(\mathrm{x}_i^2), \cdots, \mathsf{E}(\mathrm{x}_i^{\lceil d/2 \rceil}))$ and the right part is $\mathsf{E}(\vec{\mathrm{x}}_i^r) = (\mathsf{E}(\mathrm{x}_i^{\lceil d/2 \rceil + 1}), \mathsf{E}(\mathrm{x}_i^{\lceil d/2 \rceil + 2}), \cdots, \mathsf{E}(\mathrm{x}_i^d))$. Next, $\mathcal{CS}_1$ computes the sum value of these two parts

$$\mathsf{E}(\sigma_i^l) = \mathsf{E}(\sigma_i^{[1, \lceil d/2 \rceil]}) = \sum_{j=1}^{\lceil d/2 \rceil} \mathsf{E}(\mathrm{x}_i^j)$$

$$\mathsf{E}(\sigma_i^r) = \mathsf{E}(\sigma_i^{[\lceil d/2 \rceil + 1, d]}) = \sum_{j=\lceil d/2 \rceil + 1}^{d} \mathsf{E}(\mathrm{x}_i^j),$$

and generates $\langle [1, \lceil d/2 \rceil], \mathsf{E}(\sigma_1^l), \mathsf{E}(\sigma_2^l) \rangle$ and $\langle [\lceil d/2 \rceil + 1, d], \mathsf{E}(\sigma_1^r), \mathsf{E}(\sigma_2^r) \rangle$ as the root node's left and right child nodes, respectively. After choosing random numbers $r_1, r_2 \in$

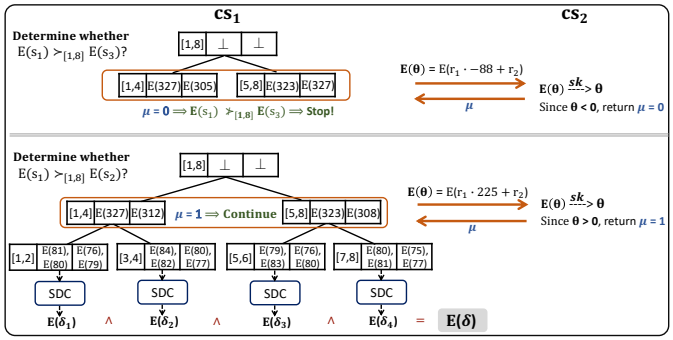



Fig. 5. Examples of DC-tree. Assume $\tau = 4$.

$\{0,1\}^{k_1}$ satisfying $r_1 > r_2 > 0$, $\mathcal{CS}_1$ computes $\mathsf{E}(\theta) = \mathsf{E}(r_1 \cdot (\sigma_2^l - \sigma_1^l) \cdot (\sigma_2^r - \sigma_1^r) + r_2)$ and sends it to $\mathcal{CS}_2$.

• Step-2. On receiving $\mathsf{E}(\theta)$, $\mathcal{CS}_2$ first uses $sk$ to recover $\theta$. If $\theta > 0$, $\mathcal{CS}_2$ makes $\mu = 1$. Otherwise $\mu = 0$. Then, $\mathcal{CS}_2$ sends $\mu$ to $\mathcal{CS}_1$.

• Step-3. If the received $\mu = 0$, $\mathcal{CS}_1$ will stop the SHDC protocol (stop building DC-tree) and make $\delta = 0$ as the output, which means $\mathsf{E}(\vec{\mathrm{x}}_1)$ does not dominate $\mathsf{E}(\vec{\mathrm{x}}_2)$. If $\mu = 1$, it indicates we cannot determine whether $\mathsf{E}(\vec{\mathrm{x}}_1)$ dominates $\mathsf{E}(\vec{\mathrm{x}}_2)$. As a result, $\mathcal{CS}_1$ makes $\mathsf{E}(\vec{\mathrm{x}}_1^l)$ and $\mathsf{E}(\vec{\mathrm{x}}_2^r)$ as new inputs and recursively performs Step-1 and Step-2. If the number of dimension of $\mathsf{E}(\vec{\mathrm{x}}_i^l)$ and $\mathsf{E}(\vec{\mathrm{x}}_i^r)$ is less than $\tau$ (usually $\tau \geq 4$), $\mathcal{CS}_1$ will not divide them and will treat them as leaf nodes: $\langle$*dimension range of* $\mathsf{E}(\vec{\mathrm{x}}_i^l), \mathsf{E}(\vec{\mathrm{x}}_1^l), \mathsf{E}(\vec{\mathrm{x}}_2^l)\rangle$ and $\langle$*dimension range of* $\mathsf{E}(\vec{\mathrm{x}}_i^r), \mathsf{E}(\vec{\mathrm{x}}_1^r), \mathsf{E}(\vec{\mathrm{x}}_2^r)\rangle$. When all leaf nodes are generated, $\mathcal{CS}_1$ performs the modified SDC protocol for each leaf node, where the inputs are the second and third fields. The modified SDC protocol does not run Step-4 and Step-5 in the original SDC protocol, and outputs $\mathsf{E}(\delta_1)$ (in Step-3 of SDC protocol) as the result. We suppose there are totally $p$ leaf nodes and denote the output of the modified SDC protocol at each leaf node as $\mathsf{E}(\delta_i')$, where $i \in [1, p]$. Afterward, $\mathcal{CS}_1$ computes $\mathsf{E}(\delta') = \mathsf{E}(\delta_1') \wedge \mathsf{E}(\delta_2') \wedge \cdots \wedge \mathsf{E}(\delta_p') = \prod_{i=1}^{p} \mathsf{E}(\delta_i')$. Next, $\mathcal{CS}_1$ obtains $\mathsf{E}(\delta'')$ by performing SBT subprotocol, in which the inputs are $\mathsf{E}(\sigma_1)$ and $\mathsf{E}(\sigma_2)$. Finally, $\mathsf{E}(\delta) = \mathsf{E}(\delta') \cdot \mathsf{E}(\delta'') = \mathsf{E}(\delta' \cdot \delta'')$ will be the output of our SHDC protocol.

Fig. 5 illustrates two examples of DC-tree. The first one is to determine whether $\mathrm{s}_1$ dominates $\mathrm{s}_3$, and the second one is to determine whether $\mathrm{s}_1$ dominates $\mathrm{s}_2$, in which $\mathrm{s}_i$ ($i = 1$ to 4) are from Fig. 1. For the first example, since one of the children of root node shows that the partial sum of $\mathrm{s}_2$ is larger than $\mathrm{s}_1$, it is definite that $\mathrm{s}_1$ does not dominates $\mathrm{s}_2$. Thus, we stop the tree building and output the result. For the second example, since $\mu = 1$, $\mathcal{CS}_1$ cannot determine whether $\mathrm{s}_1$ dominates $\mathrm{s}_2$. As the number of dimensions of the third level's nodes is less than 4, they are treated as leaf nodes and run the modified SDC protocol to determine the dominance relation.

**Correctness**. Since we assume that $\mathsf{E}(\sigma_1) \geq \mathsf{E}(\sigma_2)$, $\mathsf{E}(\vec{\mathrm{x}}_2)$ certainly does not dominate $\mathsf{E}(\vec{\mathrm{x}}_1)$. As a result, the SHDC protocol determines whether $\mathsf{E}(\vec{\mathrm{x}}_1)$ dominates $\mathsf{E}(\vec{\mathrm{x}}_2)$. We say our SHDC protocol is correct if it outputs $\mathsf{E}(1)$ when $\mathsf{E}(\vec{\mathrm{x}}_1)$ dominates $\mathsf{E}(\vec{\mathrm{x}}_2)$ and outputs $\mathsf{E}(0)$ or 0 when $\mathsf{E}(\vec{\mathrm{x}}_1)$ does not dominate $\mathsf{E}(\vec{\mathrm{x}}_2)$. First, we show that when $\delta = 0$, it is equivalent to $\mathsf{E}(\vec{\mathrm{x}}_1)$ does not dominate $\mathsf{E}(\vec{\mathrm{x}}_2)$. From Step-3, we can see that when $\mu = 0$, we have $\delta = 0$. In this case, $\theta = r_1 \cdot (\sigma_2^l - \sigma_1^l) \cdot (\sigma_2^r - \sigma_1^r) + r_2 < 0$ ($\theta$ is never equal to 0). As a result, there must exist $\sigma_2^l - \sigma_1^l > 0$ or $\sigma_2^r - \sigma_1^r > 0$. Consequently, $\exists j \in [1, d]$, $\mathsf{E}(\vec{\mathrm{x}}_2^j) > \mathsf{E}(\vec{\mathrm{x}}_1^j)$. Thus, $\mathsf{E}(\vec{\mathrm{x}}_1^j)$ does not dominate $\mathsf{E}(\vec{\mathrm{x}}_2^j)$. Then, we prove only $\mathsf{E}(\delta) = \mathsf{E}(1)$ indicates $\mathsf{E}(\vec{\mathrm{x}}_1)$ dominates $\mathsf{E}(\vec{\mathrm{x}}_2)$. Since $\mathsf{E}(\delta') = \prod_{i=1}^{p} \mathsf{E}(\delta_i')$, when $\mathsf{E}(\delta') = \mathsf{E}(1)$, it means $\forall j \in [1, d], \mathsf{E}(\vec{\mathrm{x}}_1^j) \geq \mathsf{E}(\vec{\mathrm{x}}_2^j)$. Further, $\mathsf{E}(\delta'') = \mathsf{E}(1)$ indicates $\exists j \in [1, d], \mathsf{E}(\vec{\mathrm{x}}_1^j) > \mathsf{E}(\vec{\mathrm{x}}_2^j)$. Therefore, only when $\mathsf{E}(\delta) = \mathsf{E}(1)$, $\mathsf{E}(\vec{\mathrm{x}}_1)$ dominates $\mathsf{E}(\vec{\mathrm{x}}_2)$.

In our SHDC protocol, although $\mathcal{CS}_1$ knows $\mathsf{E}(\vec{\mathrm{x}}_1)$ does not dominate $\mathsf{E}(\vec{\mathrm{x}}_2)$ in some cases, it only knows there exists one dimension $j$ that $\mathsf{E}(\vec{\mathrm{x}}_2^j) > \mathsf{E}(\vec{\mathrm{x}}_1^j)$. Since we limit the leaf node to have at least 2 ($\tau/2 > 2$) dimensions, and they will be checked as a whole in the modified SDC protocol, it guarantees the single-dimensional privacy, i.e., $\mathcal{CS}_1$ cannot infer the order relations of each dimension. Thus, our SHDC protocol can guarantee the privacy of underlying plaintexts and single-dimensional privacy. In fact, in our proposed scheme, we will use SS protocol to break the link between the sorted set and the original set so that the cloud cannot know which two data records have the no dominance relation. We will formally prove the security of our SHDC protocol in Section VI. For efficiency, our SHDC protocol will stop if the non-leaf node offers the no dominance relation, which significantly improves the performance in dealing with high-dimensional data. See Section VII-B for performance evaluations.

**Algorithm 1** Basic Skyline Computation Protocol

**Input:** An encrypted dataset $\{\mathsf{E}(\vec{\mathrm{x}}_i) \mid i \in [1, n]\}$. Assigned bit sequences for each data record, $\{b_i \mid i \in [1, n]\}$.
**Output:** A set containing encrypted skyline data records, $\mathcal{S}_{\mathrm{sky}}$.

**for** $i = 1$ to $n$ **do**
  $\mathsf{E}(\sigma_i) = \mathsf{E}(i + (n+1) \cdot \sum_{j=1}^{d} \mathrm{x}_i^j)$;
$\{\mathsf{E}(\delta_i^\rho)\}$=SS($\{\mathsf{E}(\vec{\mathrm{x}}_i, \mathsf{E}(\sigma_i))\}$), $i \in [1, n]$, $\rho \in [1, \lceil \log_2 n \rceil]$;
$\mathsf{E}(\mathrm{t}_1^j) = \sum_{i=1}^{n} \mathsf{E}(\mathrm{x}_i^j) \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil} (b_i^\rho \odot \mathsf{E}(\delta_1^\rho))$, $j \in [1, d]$;
$\mathcal{S}_{\mathrm{sky}}$.add($\mathsf{E}(\vec{\mathrm{t}}_1)$);
**for** $i = 2$ to $n$ **do**
  $\mathsf{E}(\mathrm{p}_i^j) = \sum_{l=1}^{n} \mathsf{E}(\mathrm{x}_l^j) \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil} (b_l^\rho \odot \mathsf{E}(\delta_i^\rho))$, $j \in [1, d]$;
  flag = true;
  **for** each data record $\mathsf{E}(\vec{\mathrm{t}})$ in $\mathcal{S}_{\mathrm{sky}}$ **do**
    $\delta$= WSHDC($\mathsf{E}(\vec{\mathrm{t}})$, $\mathsf{E}(\vec{\mathrm{p}}_i)$);
    **if** $\delta$ == 1 **then**
      flag = false;
      break;
  **if** flag == true **then** $\mathcal{S}_{\mathrm{sky}}$.add($\mathsf{E}(\vec{\mathrm{p}}_i)$).
**return** $\mathcal{S}_{\mathrm{sky}}$.

### *B. Privacy-Preserving Skyline Computation*

Based on the above secure protocols, we present our privacy-preserving skyline computation protocols. First, we introduce a basic protocol that is efficient but leaks single-dimensional privacy in some cases. Then, we introduce a secure protocol without leaking plaintexts, single-dimensional privacy, and access patterns.

*1) Basic Skyline Computation Protocol:* The main idea of the straw-man protocol is to leak the dominance relations to $\mathcal{CS}_1$ so that it can directly adopt the SFS algorithm to compute skyline. See Algorithm 1 for the protocol. $\mathcal{CS}_1$ first adds up

the value of all dimensions $\mathsf{E}(\sigma_i) = \mathsf{E}(\sum_{j=1}^{d} \mathrm{x}_i^j)$ for each data record. To ensure the same data record holding different sum values, $\mathcal{CS}_1$ updates $\mathsf{E}(\sigma_i)$ by $\mathsf{E}(\sigma_i) = \mathsf{E}(\sigma_i \cdot (n+1) + i)$ showing in lines 1-2. Then, $\mathcal{CS}_1$ uses the SS protocol to sort the original dataset, which not only sorts the dataset but also breaks the link between the sorted dataset and the original dataset. Next, the first data record in the sorted dataset must be a skyline point, as shown in lines 3-5. In this protocol, we weaken our SHDC protocol, denoted as WSHDC (see line 10), by returning a plaintext $\delta$ to $\mathcal{CS}_1$. In this way, $\mathcal{CS}_1$ can know $\mathsf{E}(\vec{\mathrm{x}}_1)$ dominates $\mathsf{E}(\vec{\mathrm{x}}_2)$ if $\delta = 1$, and $\mathsf{E}(\vec{\mathrm{x}}_1)$ does not dominate $\mathsf{E}(\vec{\mathrm{x}}_2)$ if $\delta = 0$. After checking dominance relation of each data record, $\mathcal{CS}_1$ obtains a skyline set, in which data records are not dominated with each other.

In the basic protocol, if there is no dominance relation between two data records, $\mathcal{CS}_1$ cannot know the actual order relation of each dimension, i.e., the basic protocol can ensure the single-dimensional privacy. Only when there is dominance relation between them, $\mathcal{CS}_1$ knows that $\forall j, \vec{\mathrm{x}}_1^j > (\text{or} \geq)\ \vec{\mathrm{x}}_2^j$, which is determined by the definition of dominance. However, the SS protocol is adopted to sort the original dataset, which makes $\mathcal{CS}_1$ learn nothing about which two data records (in original dataset) have the dominance relation.

*2) Secure Skyline Computation Protocol:* To obtain high efficiency, the basic protocol leaks single-dimensional privacy in some cases. In order to preserve this privacy, the intuitive approach is to use our SHDC protocol and output encrypted value $\mathsf{E}(\delta)$. However, it makes Algorithm 1 unavailable, since $\mathcal{CS}_1$ cannot determine whether one data record dominates the other one with the encrypted value. To tackle this issue, we design a secure skyline computation protocol that can find skylines in a secure manner, i.e., protecting underlying plaintexts, single-dimensional privacy, and access patterns. Meanwhile, in the protocol, we would like to achieve security purposes with less performance.

Algorithm 2 shows our secure skyline computation protocol. Compared with the basic protocol, this protocol uses the SHDC instead of WSHDC to check dominance relations. Consequently, there are two cases:

**Case 1**: When the non-leaf node shows there is no dominance relation between two tested data records, the SHDC protocol returns 0. If the candidate data record is not dominated by all skyline points, it will be directly added to the skyline set. See lines 13-22 for details.

**Case 2**: When all leaf nodes are checked by the modified SDC protocol, the SHDC protocol returns $\mathsf{E}(\delta)$. In this case, $\mathcal{CS}_1$ computes skyline by the fact that the data record that has the maximum sum value must be a skyline point, as shown in lines 24-30 and lines 11-12. The main idea is to maintain a dominance flag set $\mathcal{F} = \{\mathsf{E}(f_i) \mid i \in [1,n]\}$ and an encrypted sum set $\mathcal{S} = \{\mathsf{E}(\sigma_i) \mid i \in [1,n]\}$. Once the SHDC protocol outputs an encrypted dominance flag for the input data record, $\mathcal{CS}_1$ updates the following data records' (note that we have sorted the dataset by our SS protocol) dominance flags and sum values, as shown in Algorithm 3.

Update dominance flag: Suppose a data record already has a dominance flag $\mathsf{E}(f_{\text{old}})$, $f_{\text{old}} \in \{0,1\}$. For the data record, our SHDC protocol outputs an encrypted dominance flag $\mathsf{E}(\delta)$, $\delta \in \{0,1\}$. With these two encrypted data, $\mathcal{CS}_1$ updates the dominance flag as follows:

$$\begin{aligned}\mathsf{E}(f_{\text{new}}) &= \mathsf{E}(f_{\text{old}}) \vee \mathsf{E}(\delta) \\ &= \mathsf{E}(f_{\text{old}}) + \mathsf{E}(\delta) + \mathsf{E}(f_{\text{old}}) \cdot \mathsf{E}(\delta) \cdot \mathsf{E}(-1) \\ &= \mathsf{E}(f_{\text{old}} + \delta - f_{\text{old}} \cdot \delta).\end{aligned} \tag{5}$$

It indicates that once the data record is dominated by one skyline point, its final dominance flag must be $\mathsf{E}(1)$. This updating process is illustrated in lines 7-12 of Algorithm 3.

Update sum value: Suppose a data record already has an encrypted sum value $\mathsf{E}(\sigma_{\text{old}})$, and the data record's up-to-date dominance flag is $\mathsf{E}(f)$. Meanwhile, we assume $\mathcal{CS}_1$ has already computed a minimum sum value $\mathsf{E}(\texttt{MIN})$ (lines 7-8 in Algorithm 2), which guarantees that all sum values of the original dataset must be larger than MIN. First, $\mathcal{CS}_1$

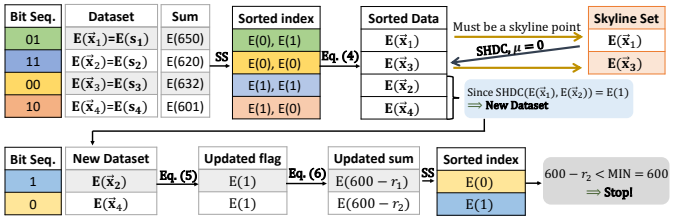



Fig. 6. Secure Skyline Computation Protocol. Assume $r_2 < r_1 \in \mathbb{Z}_n$

**Algorithm 2** Secure Skyline Computation Protocol

**Input:** An encrypted dataset $\{\mathsf{E}(\vec{\mathrm{x}}_i) \mid i \in [1,n]\}$. Assigned bit sequences for each data record, $\{b_i \mid i \in [1,n]\}$.
**Output:** A set containing encrypted skyline data records, $\mathcal{S}_{\text{sky}}$.

1: $\mathcal{X} \leftarrow \varnothing, \mathcal{S} \leftarrow \varnothing, \mathcal{F} \leftarrow \varnothing$;
2: **for** $i = 1$ to $n$ **do**
3: $\quad \mathsf{E}(\sigma_i) = \mathsf{E}(i + n + (n+1) \cdot \sum_{j=1}^{d} \mathrm{x}_i^j)$;
4: $\quad \mathcal{X}$.add($\mathsf{E}(\vec{\mathrm{x}}_i)$); $\mathcal{S}$.add($\mathsf{E}(\sigma_i)$); $\mathcal{F}$.add($\mathsf{E}(0)$);
5: $\{\mathsf{E}(\delta_i^\rho)\}$=SS($\{\mathsf{E}(\vec{\mathrm{x}}_i)\}, \{\mathsf{E}(\sigma_i)\}$), $i \in [1,n], \rho \in [1, \lceil \log_2 n \rceil]$;
6: $\mathsf{E}(\sigma_{\max}) = \sum_{i=1}^{n} \mathsf{E}(\sigma_i) \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil} (b_i^\rho \odot \mathsf{E}(\delta_1^\rho))$;
7: $\mathsf{E}(\sigma_{\min}) = \sum_{i=1}^{n} \mathsf{E}(\sigma_i) \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil} (b_i^\rho \odot \mathsf{E}(\delta_n^\rho))$;
8: $\mathsf{E}(\texttt{MIN}) = \mathsf{E}(\sigma_{\min}) + \mathsf{E}(-1) = \mathsf{E}(\sigma_{\min} - 1)$;
9: cnt = 1, N = $n$;
10: **do**
11: $\quad \mathsf{E}(\mathrm{t}_1^j) = \sum_{i=1}^{n} \mathcal{X}[i] \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil} (b_i^\rho \odot \mathsf{E}(\delta_1^\rho))$, $j \in [1,d]$;
12: $\quad \mathcal{S}_{\text{sky}}$.add($\mathsf{E}(\vec{\mathrm{t}}_1)$);
13: $\quad$ **for** $l = 2$ to $n$ **do**
14: $\quad\quad \mathsf{E}(\mathrm{p}_l^j) = \sum_{i=1}^{n} \mathcal{X}[i] \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil} (b_i^\rho \odot \mathsf{E}(\delta_l^\rho))$, $j \in [1,d]$;
15: $\quad\quad$ flag = true;
16: $\quad\quad$ **for** each data record $\mathsf{E}(\vec{\mathrm{t}})$ in $\mathcal{S}_{\text{sky}}$ **do**
17: $\quad\quad\quad \mathsf{E}(\delta)$= SHDC($\mathsf{E}(\vec{\mathrm{t}})$, $\mathsf{E}(\vec{\mathrm{p}}_l)$);
18: $\quad\quad\quad$ **if** $\mathsf{E}(\delta) \neq 0$ **then**
19: $\quad\quad\quad\quad$ flag = false;
20: $\quad\quad\quad\quad$ **break**;
21: $\quad\quad$ **if** flag == true **then**
22: $\quad\quad\quad \mathcal{S}_{\text{sky}}$.add($\mathsf{E}(\vec{\mathrm{p}}_l)$).
23: $\quad\quad$ **else**
24: $\quad\quad\quad$ Define $\{\mathsf{E}(\delta_i^\rho), b_i\}$, $i \in [1,n], \rho \in [1, \lceil \log_2 n \rceil]$ as a set $\mathcal{I}$;
25: $\quad\quad\quad$ ($\mathcal{X}$, $\mathcal{S}$, $\mathcal{F}$) = updateSets($\mathcal{X}$, $\mathcal{S}$, $\mathcal{F}$, $l$, cnt, $\mathcal{I}$, $\mathsf{E}(\texttt{MIN})$, $\mathcal{S}_{\text{sky}}$);
26: $\quad\quad\quad n = \mathcal{X}$.size, cnt = $\mathcal{S}_{\text{sky}}$.size;
27: $\quad\quad\quad$ Assign new bit sequences $\{b_i \mid i \in [1,n]\}$.
28: $\quad\quad\quad \{\mathsf{E}(\delta_i^\rho)\}$=SS($\mathcal{X}, \mathcal{S}$), $i \in [1,n], \rho \in [1, \lceil \log_2 n \rceil]$;
29: $\quad\quad\quad \mathsf{E}(\sigma_{\max}) = \sum_{i=1}^{n} \mathcal{S}[i] \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil} (b_i^\rho \odot \mathsf{E}(\delta_1^\rho))$;
30: $\quad\quad\quad$ **break**;
31: **while** BIG($\mathsf{E}(\sigma_{\max})$, $\mathsf{E}(\texttt{MIN})$) && $\mathcal{S}_{\text{sky}}$.size < N
32: **return** $\mathcal{S}_{\text{sky}}$.

**Algorithm 3** Update Sets

**Input:** Three encrypted sets, $\mathcal{X}$, $\mathcal{S}$, $\mathcal{F}$. A start point, $l$. A counter, cnt. A sorted index set, $\mathcal{I}$. An encrypted minimum sum, $\mathsf{E}(\mathtt{MIN})$. A skyline set, $\mathcal{S}_{\text{sky}}$.
**Output:** Three new encrypted sets, $\mathcal{X}_{\text{new}}, \mathcal{S}_{\text{new}}, \mathcal{F}_{\text{new}}$.
1: $\mathcal{X}_{\text{new}} \leftarrow \varnothing, \mathcal{S}_{\text{new}} \leftarrow \varnothing, \mathcal{F}_{\text{new}} \leftarrow \varnothing$, $n = \mathcal{X}$.size, $\{\mathsf{E}(\hat{\delta}_i^{\rho}), b_i\} = \mathcal{I}$;
2: **for** $k = l$ to $n$ **do**
3: $\quad \mathsf{E}(\mathtt{p}_k^j) = \sum_{i=1}^{n} \mathcal{X}[i] \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil} (b_i^{\rho} \odot \mathsf{E}(\delta_k^{\rho}))$, $j \in [1, d]$;
4: $\quad \mathsf{E}(\sigma_k) = \sum_{i=1}^{n} \mathcal{S}[i] \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil} (b_i^{\rho} \odot \mathsf{E}(\delta_k^{\rho}))$;
5: $\quad \mathsf{E}(f_k) = \sum_{i=1}^{n} \mathcal{F}[i] \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil} (b_i^{\rho} \odot \mathsf{E}(\delta_k^{\rho}))$;
6: $\quad \mathcal{X}_{\text{new}}$.add($\mathsf{E}(\vec{\mathtt{p}}_k)$), $\mathcal{S}_{\text{new}}$.add($\mathsf{E}(\sigma_k)$), $\mathcal{F}_{\text{new}}$.add($\mathsf{E}(f_k)$);
7: **for** $m =$ cnt to $\mathcal{S}_{\text{sky}}$.size **do**
8: $\quad \mathsf{E}(\vec{\mathtt{t}}_m) = \mathcal{S}_{\text{sky}}[m]$;
9: $\quad$ **for** $k = l$ to $n$ **do**
10: $\quad\quad \mathsf{E}(\delta)$= SHDC($\mathsf{E}(\vec{\mathtt{t}}_m)$, $\mathcal{X}_{\text{new}}[k - l + 1]$);
11: $\quad\quad$ **if** $\mathsf{E}(\delta) == 0$ **then** $\mathsf{E}(\delta) = \mathsf{E}(0)$;
12: $\quad\quad \mathcal{F}_{\text{new}}$.update($k - l + 1$, $\mathcal{F}_{\text{new}}[k - l + 1] \vee \mathsf{E}(\delta)$);
13: **for** $k = l$ to $n$ **do**
14: $\quad$ randomly choose $r \in \mathbb{Z}_n$ and generate $\mathsf{E}(r)$;
15: $\quad \mathsf{E}(\sigma_{\min}) = \mathsf{E}(\mathtt{MIN}) + \mathsf{E}(r) \cdot \mathsf{E}(-1) = \mathsf{E}(\mathtt{MIN} - r)$;
16: $\quad \mathsf{E}(\sigma_{\text{old}}) = \mathcal{S}_{\text{new}}[k - l + 1]$;
17: $\quad \mathsf{E}(\sigma) = \mathcal{F}_{\text{new}}[k - l + 1] \cdot (\mathsf{E}(\sigma_{\min}) - \mathsf{E}(\sigma_{\text{old}})) + \mathsf{E}(\sigma_{\text{old}})$;
18: $\quad \mathcal{S}_{\text{new}}$.update($k - l + 1$, $\mathsf{E}(\sigma)$);
19: **return** ($\mathcal{X}_{\text{new}}, \mathcal{S}_{\text{new}}, \mathcal{F}_{\text{new}}$).

chooses a random number $r \in \mathbb{Z}_n$ and generates $\mathsf{E}(r)$ by Eq. (1). Then, $\mathcal{CS}_1$ obtains a new minimum sum value: $\mathsf{E}(\sigma_{\min}) = \mathsf{E}(\mathtt{MIN}) + \mathsf{E}(r) \cdot \mathsf{E}(-1) = \mathsf{E}(\mathtt{MIN} - r)$. Next, $\mathcal{CS}_1$ updates the data record's sum value as follows:

$$\begin{aligned} \mathsf{E}(\sigma_{\text{new}}) &= \mathsf{E}(f) \cdot (\mathsf{E}(\sigma_{\min} - \sigma_{\text{old}})) + \mathsf{E}(\sigma_{\text{old}}) \\ &= \mathsf{E}(f \cdot (\sigma_{\min} - \sigma_{\text{old}}) + \sigma_{\text{old}}). \end{aligned} \tag{6}$$

If $f = 0$, the new sum value will keep the original sum value $\mathsf{E}(\sigma_{\text{new}}) = \mathsf{E}(\sigma_{\text{old}})$. If $f = 1$, $\mathsf{E}(\sigma_{\text{new}})$ will be $\mathsf{E}(\sigma_{\min})$. In this case, $\mathsf{E}(\sigma_{\text{new}})$ must be less than $\mathsf{E}(\mathtt{MIN})$ due to $\mathsf{E}(\sigma_{\min}) = \mathsf{E}(\mathtt{MIN} - r)$. Besides, since $\mathtt{MIN} > n+1$ and $r \in \mathbb{Z}_n$, $\mathtt{MIN} - r > 0$. See details of updating sum values from line 13 to line 18 in Algorithm 3.

If all sum values are updated, $\mathcal{CS}_1$ checks whether the largest sum value is bigger than $\mathsf{E}(\mathtt{MIN})$ (line 29 and line 31 in Algorithm 2). If yes, $\mathcal{CS}_1$ launches a new round to find new skyline points. Otherwise, $\mathcal{CS}_1$ stops the protocol. When all sum values are less than $\mathsf{E}(\mathtt{MIN})$, it means all data records in the current set are dominated. Here, the BIG protocol can be achieved by making $\mathcal{CS}_2$ return plaintext $\mu$ in our SBT subprotocol. Note that, since the updated values (dominates flags and sum values) usually involve several homomorphic multiplications in each round, we adopt the bootstrapping protocol (Section III-B) to refresh these ciphertexts.

Fig. 6 shows an example of our secure skyline computation protocol, in which there are four time series from Fig. 1. The sorted index are computed by our SS protocol, seeing Fig. 4 for details. In the example, $\mathsf{E}(\vec{\mathtt{x}}_1)$ must be a skyline point, as it has the largest sum value. With SHDC protocol, we can obtain that $\mathsf{E}(\vec{\mathtt{x}}_1)$ does not dominate $\mathsf{E}(\vec{\mathtt{x}}_3)$ (see the first example in Fig. 5). Therefore, $\mathsf{E}(\vec{\mathtt{x}}_3)$ is a skyline point.

### C. Description of Our Proposed Scheme

In this subsection, we present our privacy-preserving interval skyline query scheme, which is comprised of four phases: 1) system initialization; 2) data report and organization; 3) interval skyline search; and 4) data recovery.

*1) System Initialization:* In our scheme, the service provider $\mathcal{SP}$ initializes the entire system. First, given security parameters $(k_0, k_1, k_2)$, $\mathcal{SP}$ calls $\mathsf{KeyGen}(k_0, k_1, k_2)$ of SHE to generate the secret key $sk$ and $pp$. Meanwhile, $\mathcal{SP}$ generates $pk = \{\mathsf{E}(0)_1, \mathsf{E}(0)_2, pp\}$ and a public parameter $\mathsf{E}(-1)$. Then, $\mathcal{SP}$ chooses a secure hash function $\mathsf{H}()$, e.g., SHA-256, and generates two master keys $k_p, k_u$ for data providers and data users, respectively.

- When a data provider $p_i$ with its identity $\mathtt{ID}_{pi}$ registers to the system, $\mathcal{SP}$ authorizes $k_{pi} = \mathsf{H}(\mathtt{ID}_{pi}, k_p)$ to $p_i$.
- When a data user $u_i$ with the identity $\mathtt{ID}_{ui}$ registers to the system, $\mathcal{SP}$ authorizes $k_{ui} = \mathsf{H}(\mathtt{ID}_{ui}, k_u)$ to $u_i$.

Besides, since the recent data is often considered more important in time series data, it is practical to maintain the most recent timestamps. As a result, $\mathcal{SP}$ needs to determine the size $w$ (sliding window) for the most recent timestamps [5]. Finally, $\mathcal{SP}$ publishes $\{\mathsf{H}(), w, pk, \mathsf{E}(-1)\}$, sends $\{k_p, k_u\}$ to $\mathcal{CS}_1$, and authorizes $sk$ to $\mathcal{CS}_2$.

*2) Data Report and Organization:* If a data provider $p_i$ has a sensed data $\mathtt{x}_i^j$ at a time stamp $t_j$, it will encrypt $\mathtt{x}_i^j$ into $\mathsf{E}(\mathtt{x}_i^j)$ by using Eq. (1). Then, $p_i$ computes $\mathsf{H}(\mathsf{E}(\mathtt{x}_i^j), k_{pi})$ with the authorized key $k_{pi}$ and sends $\langle p_i, t_j, \mathsf{E}(\mathtt{x}_i^j)\rangle || \mathtt{ID}_{pi} || \mathsf{H}(\mathsf{E}(\mathtt{x}_i^j), k_{pi})$ to $\mathcal{CS}_1$.

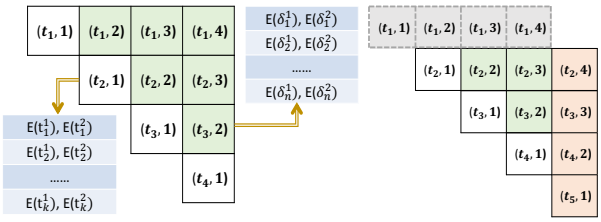
Upon receiving it, $\mathcal{CS}_1$ first computes $k_{pi} = \mathsf{H}(\mathtt{ID}_{pi}, k_p)$ with the authorized $k_p$ and then calculates $\mathsf{H}(\mathsf{E}(\mathtt{x}_i^j), k_{pi})$. Next, $\mathcal{CS}_1$ extracts $\mathsf{H}(\mathsf{E}(\mathtt{x}_i^j), k_{pi})$ from the received message and checks whether the calculated $\mathsf{H}(\mathsf{E}(\mathtt{x}_i^j), k_{pi})$ is the same as the extracted one. If yes, $\mathcal{CS}_1$ accepts the reported data, otherwise rejects. Afterward, $\mathcal{CS}_1$ randomly generates a unique bit sequence $\{b_i = b_i^1 b_i^2 \cdots b_i^{\rho} \cdots b_i^{\lceil \log_2 n \rceil}\}$ for $p_i$, where $n$ is the number of data providers, and $b_i^{\rho} \in \{0, 1\}$. Finally, $\mathcal{CS}_1$ organizes the reported time series data as follows.

TABLE II
TIME SERIES DATA

| | Bit Seq. | $t_1$ | $t_2$ | ... | $t_j$ | ... | $t_w$ |
|---|---|---|---|---|---|---|---|
| $p_1$ | $b_1$ | $\mathsf{E}(\mathtt{x}_1^1)$ | $\mathsf{E}(\mathtt{x}_1^2)$ | ... | $\mathsf{E}(\mathtt{x}_1^j)$ | ... | $\mathsf{E}(\mathtt{x}_1^w)$ |
| $p_2$ | $b_2$ | $\mathsf{E}(\mathtt{x}_2^1)$ | $\mathsf{E}(\mathtt{x}_2^2)$ | ... | $\mathsf{E}(\mathtt{x}_2^j)$ | ... | $\mathsf{E}(\mathtt{x}_2^w)$ |
| $p_i$ | $b_i$ | $\mathsf{E}(\mathtt{x}_i^1)$ | $\mathsf{E}(\mathtt{x}_i^2)$ | ... | $\mathsf{E}(\mathtt{x}_i^j)$ | ... | $\mathsf{E}(\mathtt{x}_i^w)$ |

If a new data is reported, the data in $t_1$ column will be expired, and a new column $t_{w+1}$ will be added into Table II. Note that, similar to [5], we assume all time series are synchronized.

*3) Interval Skyline Search:* When a data user $u_i$ would like to use the interval skyline query services, he/she can first define a time interval $[t_{j_1} : t_{j_2}] \subset [t_1 : t_w]$, $t_{j_1} < t_{j_2}$, and computes $\mathsf{H}(t_{j_1} || t_{j_2}, k_{ui})$ with the authorized key $k_{ui}$. Then, $u_i$ sends $[t_{j_1} : t_{j_2}] || \mathtt{ID}_{ui} || \mathsf{H}(t_{j_1} || t_{j_2}, k_{ui})$ to $\mathcal{CS}_1$. Using the same approach introduced in the data report and organization phase, $\mathcal{CS}_1$ checks whether $\mathsf{H}(t_{j_1} || t_{j_2}, k_{ui})$ is correct. If yes, $\mathcal{CS}_1$ first extracts the encrypted data from $t_{j_1}$ to $t_{j_2}$ for all data provides and generates an encrypted dataset $\{\mathsf{E}(\vec{\mathtt{x}}_i) = (\mathsf{E}(\mathtt{x}_i^1), \mathsf{E}(\mathtt{x}_i^2), \cdots, \mathsf{E}(\mathtt{x}_i^{j_2 - j_1 + 1})) \mid i \in [1, n]\}$. Then, $\mathcal{CS}_1$ runs




(a) 2-dimensional table (b) Update

Fig. 7. 2-dimensional look-up table

our secure skyline computation protocol (Section V-B2) to obtain the desired interval skyline points $\{\mathsf{E}(\vec{t}_i) \mid i \in [1,k]\}$, where $k$ is the number of skyline points.

To quickly respond to the interval skyline query, we design a 2-dimensional look-up table, as shown in Fig. 7(a), to index the interval skyline points. In Fig. 7(a), we assume the sliding window $w = 5$ ranges from $t_1$ to $t_5$. First, $\mathcal{CS}_1$ generates the 2-dimensional table, in which each cell is a pair: $\langle$*start time*, *time step*$\rangle$. For example, $\langle t_2, 3\rangle$ indicates the time interval $[t_2, t_5]$. If a data user launches an interval skyline query $[t_2, t_5]$, $\mathcal{CS}_1$ will store the computed results under the cell after responding to the query:

- If the time step is less than $\log_2 n$, $\mathcal{CS}_1$ directly stores the computed skyline set $\mathcal{S}_{\text{sky}}$ under the cell.
- If the time step is larger than or equal to $\log_2 n$, $\mathcal{CS}_1$ stores the sorted index $\{\mathsf{E}(\delta_i^\rho)\}$, $i \in [1,n], \rho \in [1, \lceil \log_2 n \rceil]$.

This strategy can balance the storage costs and computational costs. When another query request falls in the cell (the time complexity of locating cell is O(1)), $\mathcal{CS}_1$ can directly obtain the interval skyline or skip the SS protocol to compute the interval skyline. In the example of Fig. 7(a), if we suppose there are 4 time series, i.e., $n = 4$, all the cells, whose time step is 1, will store the skyline set. Others will store the sorted index. Meanwhile, it is simple and efficient for the 2-dimensional table to support the continuous updates over time. Fig. 7(b) shows that $t_1$ is expired, while $t_6$ is added. It is worth noting that only the cell (time interval) has been queried, the computed results would be stored under the cell. Otherwise, the cell is empty. This mechanism is built according to the following facts: i) some cells may be queried frequently. ii) some cells may be expired before being queried.

After obtaining the encrypted skyline set, $\mathcal{CS}_1$ will generate a random number $r$ and compute $\{r_i^j = \mathsf{H}(i||j,r) \mid i \in [1,k], j \in [1,d], r_i^j \in \{0,1\}^{k_1}\}$. Then, $\mathcal{CS}_1$ adds these random noises to the corresponding skyline point: $\mathsf{E}(\mathtt{t}_i^j + r_i^j)$. Next, $\mathcal{CS}_1$ transfers $\{\mathsf{E}(\mathtt{t}_i^j + r_i^j) \mid i \in [1,k], j \in [1,d]\}$ to $\mathcal{CS}_2$ and sends $r$ to the data user $u_i$ via a secure channel. For $\mathcal{CS}_2$, it will recover $\{\mathtt{t}_i^j + r_i^j \mid i \in [1,k], j \in [1,d]\}$ with the secret key $sk$ of SHE and returns them to $u_i$.

*4) Data Recovery:* Upon receiving $r$ from $\mathcal{CS}_1$ and $\{\mathtt{t}_i^j + r_i^j \mid i \in [1,k], j \in [1,d]\}$ from $\mathcal{CS}_2$, $u_i$ first computes $\{r_i^j = \mathsf{H}(i||j,r) \mid i \in [1,k], j \in [1,d]\}$. Then, $u_i$ removes the random noises $r_i^j$ from $\{\mathtt{t}_i^j + r_i^j \mid i \in [1,k], j \in [1,d]\}$. Finally, $u_i$ can obtain the skyline point $\{\mathtt{t}_i^j \mid i \in [1,k], j \in [1,d]\}$ of the time interval $[t_{j_1} : t_{j_2}]$. This approach does not need to authorize $sk$ to data users for recovering plaintexts, and thus avoids the risk brought from over authorization.

## VI. Security Analysis

In this section, we analyze the security of our proposed scheme. Since the core part of our proposed scheme is to perform the interval skyline search, we will prove that our secure skyline computation protocol can protect the plaintext of data records and query results, single-dimensional privacy, and access patterns. Before analyzing the security of our scheme, we first introduce a composition theorem.

**Theorem 1.** ***(Composition Theorem) [9], [19].*** *If a protocol consists of several subprotocols, the protocol is secure as long as the subprotocols are secure and all the intermediate results are random or pseudo-random.*

We refer readers to [19] for the detailed proof. In Fig. 8, we present the structure of our secure skyline computation protocol. It is clear that we should first prove that the SBT and SEQ subprotocols are secure, which can demonstrate the security of SDC and SHDC protocols according to Theorem 1. After analyzing the security of SS, SDC, and SHDC, we can obtain the security of our secure skyline computation protocol.

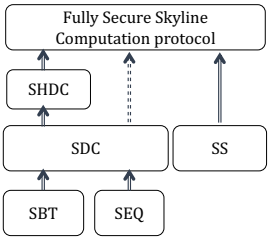



Fig. 8. Structure of our proposed scheme

Next, we briefly review the security model for securely realizing an ideal functionality in the presence of non-colluding semi-honest adversaries [19], [20].

**Real-world execution.** The real-world execution of a protocol $\Pi$ takes place between $\{\mathcal{CS}_1, \mathcal{CS}_2\}$ and adversaries $\{\mathcal{A}_1, \mathcal{A}_2\}$, who corrupt $\mathcal{CS}_1$ and $\mathcal{CS}_2$, respectively. With the input $x_i$ and auxiliary input $z_i$ $(i = 1, 2)$, e.g., the length of ciphertexts, the view of each party in the real-world execution protocol $\Pi$ in the presence of adversary $\mathcal{A}_1$ ($\mathcal{A}_2$) is defined as

$\text{REAL}_{\Pi,\mathcal{A}_i,z_i}(x_i) \stackrel{def}{=} \text{OUT}_i^{\Pi}, i = 1, 2.$

**Ideal-world execution.** In the ideal world execution, there is an ideal functionality $\mathcal{F}$ for a function $f$, and the servers interact only with $\mathcal{F}$. Here, the view of each party in an ideal-world execution in the presence of independent simulators $\{\mathtt{Sim}_1, \mathtt{Sim}_2\}$ is defined as

$\text{IDEAL}_{\mathcal{F},\mathtt{Sim}_i,z_i}(x_i) \stackrel{def}{=} \text{OUT}_i^{\mathcal{F}}, i = 1, 2.$

In the above definitions, $\text{OUT}_i$ denotes the output of parties.

**Definition 4** (Security against semi-honest adversaries)**.** *Let $\mathcal{F}$ be a deterministic functionality and $\Pi$ be a protocol between two parties (servers). We say that $\Pi$ securely realizes $\mathcal{F}$ if there exists $\{\mathtt{Sim}_1, \mathtt{Sim}_2\}$ of PPT (Probabilistic Polynomial Time) transformations (where $\mathtt{Sim}_i = \mathtt{Sim}_i(\mathcal{A}_i), i = 1, 2$) such that for semi-honest PPT adversaries $\{\mathcal{A}_1, \mathcal{A}_2\}$, for all $x_i$ and $z_i$, for each party holds:*

$$\mathit{REAL}_{\Pi,\mathcal{A}_i,z_i}(x_i) \stackrel{c}{\approx} \mathit{IDEAL}_{\mathcal{F},\mathtt{Sim}_i,z_i}(x_i)$$

*where $\overset{c}{\approx}$ compactly denotes computational indistinguishability.*

### A. The SBT and SEQ subprotocols are privacy-preserving.

First, we use Definition 4 to show that our SBT subprotocol is secure, i.e., it can preserve the privacy of plaintexts.

**Theorem 2.** *The SBT subprotocol securely determines the bigger than relation in the presence of semi-honest (non-colluding) adversaries $\{\mathcal{A}_1, \mathcal{A}_2\}$.*

*Proof.* Here, we show how to construct the independent simulators: $\{\mathtt{Sim}_1, \mathtt{Sim}_2\}$.

$\mathtt{Sim}_1$ randomly chooses $\{m_1', m_2', \delta'\}$ and simulates $\mathcal{A}_1$ as follows. It first generates ciphertexts $\{\mathsf{E}(m_1'), \mathsf{E}(m_2'), \mathsf{E}(\delta')\}$ by the encryption algorithm of SHE. Then, it outputs $\mathcal{A}_1$'s entire view. In the real execution, $\mathcal{A}_1$ receives the ciphertext of $\{m_1, m_2, \delta\}$ whereas $\mathtt{Sim}_1$ gives $\{\mathsf{E}(m_1'), \mathsf{E}(m_2'), \mathsf{E}(\delta')\}$ to $\mathcal{A}_1$ in the ideal execution. The semantic security of SHE [17] guarantees that the views of $\mathcal{A}_1$ in the real and the ideal executions are indistinguishable.

$\mathtt{Sim}_2$ runs $\mathcal{A}_2$ by randomly choosing $\theta'$ and sending it to $\mathcal{A}_2$, which is the view of $\mathcal{A}_2$ in the ideal execution. In the real execution, $\mathcal{A}_2$ receives $\theta = s \cdot r_1 \cdot (m_1 - m_2) - s \cdot r_2$. Since $r_1, r_2$ and $s$ are randomly generated, the views of $\mathcal{A}_2$ in the real and the ideal executions are indistinguishable. Note that, since $\theta'$ is randomly chosen, it means $\mu'$ will also be randomly generated. $\mathcal{A}_2$ cannot distinguish the views in $\mu'$ and $\mu$. □

From the above proof, we can see that our SBT subprotocol ensures the security of the plaintext of $\{m_1, m_2\}$ and the result $\delta$, i.e., preserving the privacy of inputs and order relations. In our SBT subprotocol, since $r_2 < r_1$, and $s$ is randomly generated, $\mathcal{CS}_2$ cannot infer whether $m_1 = m_2$ when $m_1 = m_2$. Therefore, our SBT subprotocol is privacy-preserving.

**Theorem 3.** *The SEQ subprotocol securely determines the equality relation in the presence of semi-honest (non-colluding) adversaries $\{\mathcal{A}_1, \mathcal{A}_2\}$.*

*Proof.* Similar to the proof of Theorem 2, we can adopt Definition 4 to show that our SEQ subprotocol is secure. That is, it can preserve the privacy of plaintexts and equality relations. □

### B. The SS, SDC, and SHDC protocols are privacy-preserving.

First, we show that our SS protocol is secure, i.e., it can preserve the privacy of plaintexts.

**Theorem 4.** *The SS protocol securely sorts $\{\mathsf{E}(\vec{\mathtt{x}}_i) = (\mathsf{E}(\mathtt{x}_i^1), \mathsf{E}(\mathtt{x}_i^2), \cdots, \mathsf{E}(\mathtt{x}_i^d)) \mid \mathtt{x}_i^j \in \mathcal{M}, i \in [1, n], j \in [1, d]\}$ in the presence of semi-honest (non-colluding) adversaries $\{\mathcal{A}_1, \mathcal{A}_2\}$.*

*Proof.* Since $\mathcal{CS}_1$ always processes data records over their ciphertexts, $\mathcal{A}_1$ receives encrypted data in both views. Similar to $\mathcal{A}_1$'s views in the SBT subprotocol, the semantic security of SHE guarantees that the views of $\mathcal{A}_1$ in the real and the ideal executions are indistinguishable.

$\mathtt{Sim}_2$ runs $\mathcal{A}_2$ as follows. First, $\mathtt{Sim}_2$ randomly chooses $\{\vec{\mathtt{x}'}_i = (\mathtt{x}'^1_i, \mathtt{x}'^2_i, \cdots, \mathtt{x}'^d_i) \mid \mathtt{x}'^j_i \in \mathcal{M}, i \in [1, n], j \in [1, d]\}$, and calculates $\sigma_i' = \sum_{j=1}^{d} \mathtt{x}'^j_i, i \in [1, n]$. After choosing $n+1$ random numbers $\{r_0, r_1, \cdots, r_n\}$ satisfying $r_0 > r_i, i \in [1, n]$, it computes $\{\hat{\sigma}_i' = r_0 \cdot \sigma_i' + r_i \mid i \in [1, n]\}$ and sends them to $\mathcal{A}_2$, which is the view of $\mathcal{A}_2$ in the ideal execution. In the real execution, $\mathcal{A}_2$ receives $\{\hat{\sigma}_i = r_0 \cdot \sigma_i + r_i \mid i \in [1, n]\}$, where $\sigma_i = \sum_{j=1}^{d} \mathtt{x}_i^j$. Since both $\mathtt{x}_i^j$ and $\mathtt{x}'^j_i \in \mathcal{M}$, and $n+1$ random numbers are chosen to perturb $\sigma_i$ and $\sigma_i'$, the views of $\mathcal{A}_2$ in the real and the ideal executions are indistinguishable. □

The above proof shows that our SS protocol can preserve the privacy of plaintexts $\{\vec{\mathtt{x}}_i = (\mathtt{x}_i^1, \mathtt{x}_i^2, \cdots, \mathtt{x}_i^d) \mid \mathtt{x}_i^j \in \mathcal{M}, i \in [1, n], j \in [1, d]\}$ and results $\{\mathsf{E}(\hat{\delta}_i^\rho) \mid 1 \le \rho \le \lceil \log_2 n \rceil, i \in [1, n]\}$ (proved in $\mathcal{A}_1$'s views).

Next, we prove that our SDC and SHDC protocols are secure, i.e., they can protect inputs, results, and single-dimensional privacy.

**Theorem 5.** *The SDC and SDHC protocols securely determine dominance relations without leaking inputs, the protocol results, and the single-dimensional privacy.*

*Proof.* SDC: As shown in Fig. 8, the SDC protocol consists of the SBT and SEQ subprotocols. Recalling Section V-A2, before running SEQ and SBT subprotocols, the SDC protocol generates $\{\theta^i \mid i \in [1, d]\}$, $\alpha$, and $\beta$ (see Section V-A2). Since all of these values are random, and SBT and SEQ subprotocols are secure (Theorem 2 and Theorem 3 ), we can prove that our SDC protocol is secure according to the composition theorem given in Theorem 1. That is, our SDC protocol can protect the plaintext of data records and protocol result from leaking. Besides, our SDC protocol uses the flip-coin mechanism to randomly adjust the order relation of two values. Therefore, it can protect the single-dimensional privacy, i.e, the order or equality relation of each dimension's values. Thus, our SDC protocol is privacy-preserving.

SHDC: If our SHDC protocol determines the dominance relation of two data records by leaf nodes, it has the same security as the SDC protocol. On the other hand, if the dominance relation is determined by non-leaf nodes, $\mathcal{CS}_1$ and $\mathcal{CS}_2$ can know whether one data record does not dominate the other one. It is a trivial leakage, denoted as $\mathcal{L} = \{\mathtt{x}_{i_1} \not\succ \mathtt{x}_{i_2} \mid i_1 \neq i_2\}$, because $\mathcal{CS}_1$ and $\mathcal{CS}_2$ only know there exists one dimension $j \in [1, d]$ that $\mathtt{x}_{i_1}^j < \mathtt{x}_{i_2}^j$. Since neither $\mathcal{CS}_1$ nor $\mathcal{CS}_2$ knows which dimension has such the order relation, the single-dimensional privacy is protected in our SHDC protocol. Thus, our SHDC protocol is privacy-preserving. □

### C. The secure skyline computation protocol is privacy-preserving.

In this subsection, we first prove that our secure skyline computation protocol can preserve the privacy of inpputs, the protocol results, and the single-dimensional privacy. Then, we show that our secure skyline computation protocol can hide access patterns.

**Theorem 6.** *The secure skyline computation protocol securely computes skyline points without leaking inputs, the protocol results, and the single-dimensional privacy.*

*Proof.* From Algorithm 2, we can see that all of the intermediate results are random. According to Theorem 1, our secure skyline computation protocol can ensure the plaintext of data records and protocol results without leaking. For single-dimensional privacy, it only involves the SHDC protocol. Although there is a trivial leakage $\mathcal{L}$, we have proved that the single-dimensional privacy is preserved in our SHDC protocol. Therefore, our secure skyline computation protocol can preserve single-dimensional privacy. □

**Theorem 7.** *The secure skyline computation protocol can hide the information about which data records in* $\{\mathsf{E}(\vec{\mathbf{x}}_i) \mid i \in [1,n]\}$ *are selected as skyline points.*

*Proof.* We prove Theorem 7 by demonstrating neither $\mathcal{CS}_1$ nor $\mathcal{CS}_2$ knows which data records are selected as skyline points.

For $\mathcal{CS}_1$. From Algorithm 2, there are two cases for a data record to be added into the skyline set.

Case1: A data record that is not dominated by any skyline point is a skyline point (line 22 in Algorithm 2). Since our SS protocol returns the encrypted sort index $\{\mathsf{E}(\delta_i^\rho)\}$, and the data record is computed from $\sum_{l=1}^{n} \mathsf{E}(\mathbf{x}_l^j) \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil}(b_l^\rho \odot \mathsf{E}(\delta_i^\rho)), j \in [1,d], i \in [1,n]$, $\mathcal{CS}_1$ cannot link the computed data record to the data record in the dataset $\{\mathsf{E}(\vec{\mathbf{x}}_i) = (\mathsf{E}(\mathbf{x}_i^1), \mathsf{E}(\mathbf{x}_i^2), \cdots, \mathsf{E}(\mathbf{x}_i^d)) \mid i \in [1,n]\}$. Therefore, $\mathcal{CS}_1$ does not know which item is selected as skyline point in this case.

Case2: A data record that has the maximum sum value is a skyline point (line 12 in Algorithm 2). Similarly, the data point is calculated by $\sum_{l=1}^{n} \mathsf{E}(\mathbf{x}_l^j) \cdot \prod_{\rho=1}^{\lceil \log_2 n \rceil}(b_l^\rho \odot \mathsf{E}(\delta_1^\rho)), j \in [1,d]$ (line 11). Consequently, $\mathcal{CS}_1$ does not know which item is selected as skyline point in this case.

For $\mathcal{CS}_2$. From Algorithm 2, we can see that $\mathcal{CS}_2$ can get information when running the SS and SHDC protocols.

SS: $\mathcal{CS}_2$ obtains the perturbed sum value of each data record. Besides, before sending encrypted sum values to $\mathcal{CS}_2$, $\mathcal{CS}_1$ randomly permutes them with XOR gate. Therefore, $\mathcal{CS}_2$ cannot link these sum values to the corresponding data record.

SHDC: For the non-leaf nodes, $\mathcal{CS}_2$ can learn the information about *one data record does not dominate the other one* by observing whether $\theta = r_1 \cdot (\sigma_2^l - \sigma_1^l) \cdot (\sigma_2^r - \sigma_1^r) + r_2 > 0$, where $\sigma_i^l$ and $\sigma_i^r$ $(i = 1, 2)$ are partial sum values (see details in Section V-A3). However, $\mathcal{CS}_2$ cannot link the partial sum values to the corresponding data record due to the perturbation and permutation. When reaching leaf nodes, our SHDC protocol will invoke the SDC protocol. However, Theorem 5 shows that $\mathcal{CS}_2$ learns nothing from the SDC protocol. Therefore, $\mathcal{CS}_2$ does not know which items are selected as skyline points. □

## VII. Performance Evaluation

In this section, we evaluate the performance of our proposed scheme. From Section V-C, we know that the data report and data recovery phases only involve data encryption and noise removal, respectively, which are efficient and intuitive in terms of performance. Therefore, in this section, we focus on the performance of the interval skyline search phase.

Experimental setting: We implemented our scheme in Java and executed it on a machine with 16 GB memory, 3.4 GHz Intel(R) Core(TM) i7-3770 processors, and Ubuntu 16.04 OS. In our experiments, we adopt two real-world time series datasets and illustrate the detailed information in Table III. For simplicity sake, we denote these two datasets as **Gas** and **Electricity**, respectively. It is worth noting there are around 100,000 time stamps in the **Electricity** dataset. We filter out the missing values (the value of two consecutive time stamps is 0), and the effective timestamps are 10,303.

TABLE III
Real-world Datasets Used in Our Experiments

| **Name** | **Time series** ($n$) | **Timestamps** ($d$) |
|---|---|---|
| Greenhouse Gas Observing Network Data Set [21] | 2912 | 327 |
| Electricity Load Diagrams Data Set [22] | 315 | 10303 |

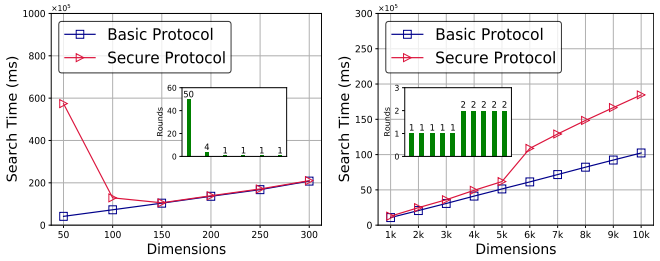
To ensure privacy, we set $\tau = 4$ in our SHDC protocol, i.e., the number of dimensions of non-leaf nodes should be larger than 4. For security parameters of SHE, we let $k_0 = 4096, k_1 = 40, k_2 = 160$ in our evaluations.

### A. Performance of Our Proposed Scheme

Since the performance of our skyline computation protocols is related to the number of time series $n$ and timestamps (dimensions) $d$, in this subsection, we evaluate the computational cost and the communication cost of the basic and secure skyline computation protocols by varying these two parameters on **Gas** and **Electricity** datasets. Here, we respectively denote these two protocols as basic protocol and secure protocol in the following discussion.

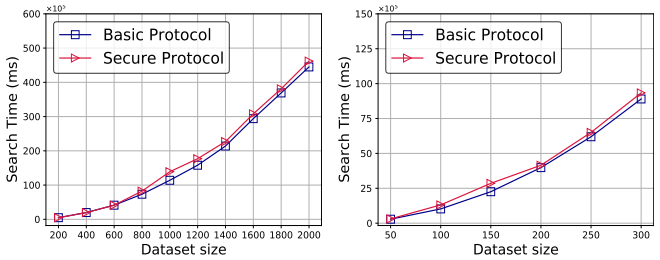
• *Computational cost of searching interval skyline.* Fig. 9 depicts the search time varying with $d$, in which Fig. 9(a) shows the evaluation over **Gas** dataset and $n = 800$, while Fig. 9(b) shows the evaluation over **Electricity** dataset and $n = 100$. In Fig. 9(a), the search time of the secure protocol is more than the basic protocol when $d$ is small. It is reasonable since, in the **Gas** dataset, the number of skyline points is not changed when we vary $d$. When $d$ is small, it means that $\mathcal{CS}_1$ has a high probability of obtaining the dominance relations from the leaf nodes, leading to more rounds to get the whole skyline points. However, when $d$ is large, $\mathcal{CS}_1$ may get the dominance relation from non-leaf nodes, leading to fewer rounds. When the number of rounds is 1, the computational cost of the secure protocol is equivalent to using the SFS algorithm and has a similar search time to the basic protocol. The trend of Fig. 9(b) is also related to the number of rounds, which depends on the dataset. To clearly show the reason, we plot the number of rounds in the corresponding reduced figure.

Fig. 10 plots the search time varying with $n$, in which Fig. 10(a) shows the evaluation over **Gas** dataset and $d = 100$, while Fig. 10(b) shows the evaluation over **Electricity** dataset and $d = 1000$. In both figures, since the secure protocol



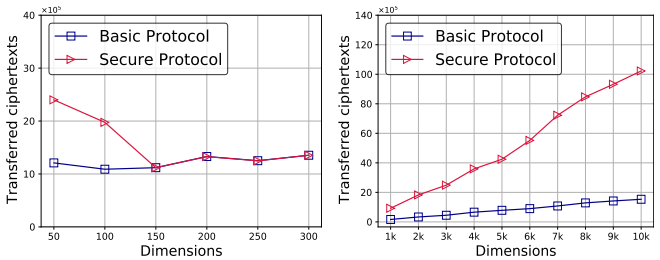

(a) **Gas** (b) **Electricity**

Fig. 9. Computational costs of searching interval skyline varying with the number of dimensions. (a) Over **Gas** dataset; (b) Over **Electricity** dataset.



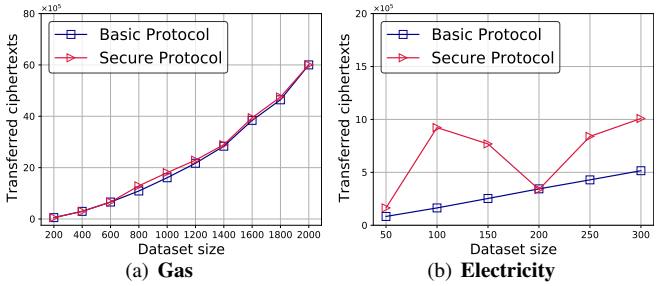

(a) **Gas** (b) **Electricity**

Fig. 10. Computational costs of searching interval skyline varying with the number of data records. (a) Over **Gas** dataset; (b) Over **Electricity** dataset.




(a) **Gas** (b) **Electricity**

Fig. 11. Communication costs of searching interval skyline varying with the number of dimensions. (a) Over **Gas** dataset; (b) Over **Electricity** dataset.




(a) **Gas** (b) **Electricity**

Fig. 12. Communication costs of searching interval skyline varying with the number of data records. (a) Over **Gas** dataset; (b) Over **Electricity** dataset.

computes the skyline points with one round in the most cases, it means that dominance relations are determined by the non-leaf nodes of our SHDC protocol. Therefore, our secure protocol has a similar search time as the basic protocol.

• *Communication cost of searching interval skyline.* Fig. 11 and Fig. 12 depict the communication cost of the secure and basic protocols, in which we vary $n$ and $d$, respectively. For simplicity sake, we measure the communication cost by the number of ciphertexts transferred between two servers: $\mathcal{CS}_1$ and $\mathcal{CS}_2$. Fig. 11(a) and Fig. 11(b) show the similar trends to Fig. 9(a) and Fig. 9(b), respectively. That is because the communication cost is also related to the number of rounds. However, Fig. 12 shows a different trend. In Fig. 12(a), since the numbers of skyline points and rounds are not changed when $n$ is increasing over the **Gas** dataset , the communication cost of the secure and basic protocol is close. In Fig. 12(b), the secure protocol has more communication cost than the basic protocol. That is because the number of skyline points $k$ would affect the communication costs if $k$ is relatively small compared to the whole dataset. In our protocols, each skyline point is used to check the dominance relation with non-skyline points. Therefore, the more skyline points will result in the more communication costs. In the experiment of Fig. 12(b), the number of skyline points is $\{2, 6, 4, 1, 2, 2\}$ for the dataset size from 50 to 300. Thus, the trend in Fig. 12(b) is reasonable. Note that, when $n$ is 200, the number of skyline points is 1. In this case, the communication cost of the secure protocol should be similar to that of basic protocol (See Algorithms 1 and 2). Fig. 12(b) demonstrates the correctness of our analysis.

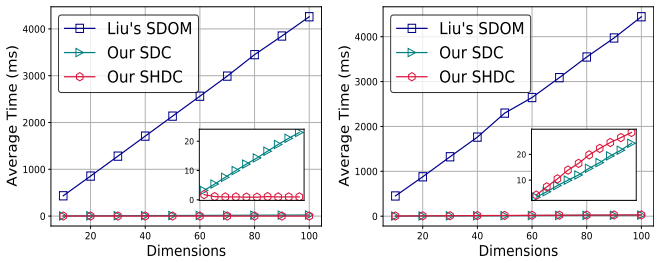
### *B. Comparing Dominance Check Protocols*

In the process of skyline search, the core component is the secure dominance check protocol that can determine whether two encrypted data records have a dominance relation. In this paper, we propose two secure dominance check protocols, SDC and SHDC, in which SDC is the general version, and SHDC is for high-dimensional data records. Both of them can preserve the privacy of plaintexts and single-dimensional privacy. In [9], Liu et al. also proposed a secure dominance check protocol, hereafter we denote it as Liu's SDOM, which has the same security level as ours and can be used in our scheme. As a result, in this subsection, we will compare our SDC, SHDC, and Liu's SDOM protocols in terms of computational and communication costs.

However, Liu's SDOM protocol employs the Paillier encryption, which is more expensive than SHE scheme used in our protocols. Thus, to be fair, we implemented Liu's SDOM protocol with SHE and compare these protocols based on the same cryptographic primitive.

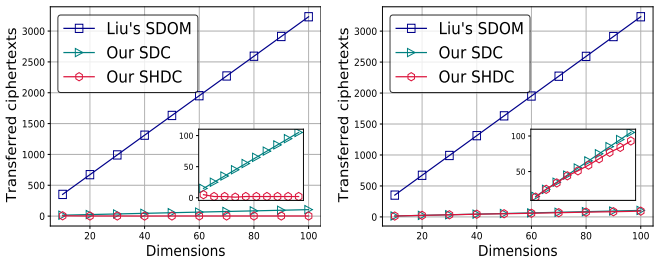
However, Liu's SDOM protocol employs a different homomorphic encryption scheme, namely, Paillier encryption. In Table IV, we list the computational cost of these two cryptosystems in the operations of encryption (Enc), decryption (Dec), homomorphic addition (Homo-Add), and homomorphic multiplication (Homo-Mul ) varying the security parameter $k = |q| = |p|$ from 512 to 2048. From Table IV, we can see that the Paillier encryption is more expensive than the SHE scheme used in our protocols. Thus, to be fair, we implemented Liu's SDOM protocol with SHE and compare these protocols based on the same cryptographic primitive.

• *Comparing computational costs.* From Section V-A2, V-A3, and [9], we know that the computational cost of these three protocols is only related to the number of dimensions. Accordingly, Fig. 13 depicts average execution costs of our




(a) **Gas** (b) **Electricity**

Fig. 13. Computational costs of secure dominance check protocols. (a) Over **Gas** dataset; (b) Over **Electricity** dataset.




(a) **Gas** (b) **Electricity**

Fig. 14. Communication costs of secure dominance check protocols. (a) Over **Gas** dataset; (b) Over **Electricity** dataset.

TABLE IV
THE EXECUTION TIME OF SHE AND PAILLIER

| $k$ | Scheme | Enc | Dec | Homo-Add | Homo-Mul |
|---|---|---|---|---|---|
| 512 | SHE | 0.03 ms | 0.003 ms | 0.002 ms | 0.002 ms |
| | Paillier | 2.6 ms | 2.4 ms | 0.02 ms | 0.05 ms |
| 1024 | SHE | 0.04 ms | 0.004 ms | 0.004 ms | 0.005 ms |
| | Paillier | 18 ms | 17 ms | 0.05 ms | 0.17 ms |
| 2048 | SHE | 0.8 ms | 0.01 ms | 0.01 ms | 0.01 ms |
| | Paillier | 142 ms | 142 ms | 0.12 ms | 0.62 ms |

SDC, our SHDC, and Liu's SDOM varying with the number of dimensions from 10 to 100. Fig. 13(a) and 13(b) show the evaluations over **Gas** and **Electricity** datasets, respectively. Both of figures show that our protocols can improve the efficiency of determining dominance relation by at least two orders of magnitude compared to Liu's SDOM protocol. That is because: i) the secure comparison subprotocol (SBT) used in our protocols is more efficient than that of Liu's SDOM; ii) our designed algorithm is more efficient. We determine whether $\forall j \in [1, d]$ $\mathbf{x}_1^j \geq \mathbf{x}_2^j$ by testing whether $\alpha = \beta$ (see details in Section V-A2), while Liu's SDOM protocol checks all dimensions one by one.

It is interesting that the computational cost of our SHDC protocol is much less than our SDC protocol in Fig. 13(a), whereas it is slightly more computationally expensive in Fig. 13(b). This is reasonable since most of the data records in the **Gas** dataset have no dominance relation with each other and thus make our SHDC protocol determine the dominance relation by the non-leaf nodes of DC-tree instead of leaf nodes. It can greatly improve performance and, it is our intention to design such a high-dimensional protocol. While for the **Electricity** dataset, there exist several data records that can dominate others. As a result, our SHDC protocol determines dominance relations by leaf nodes of DC-tree, i.e., invoking SDC protocol, in most cases. Therefore, the computational cost of SHDC protocol is approximately equivalent to *time cost of checking non-leaf nodes* + *time cost of SDC protocol*. Thus, for the **Electricity** dataset, our SHDC protocol takes more time than our SDC protocol.

• *Comparing communication costs.* Fig. 14 plots the number of transferred ciphertexts of these three protocols varying with the number of dimensions, in which Fig. 14(a) shows the communication cost over the **Gas** dataset, while Fig. 14(b) is over the **Electricity** dataset. Both figures show that our protocols entail at least 23× improvement in the communication cost. The reason is that it needs $d + 5$ ciphertexts to achieve our SDC protocol, while it is $2(d + 1)(l + 1)$ for Liu's SDOM protocol, where $d$ is the number of dimensions and $l$ is the largest bit length of values in the evaluated dataset. Meanwhile, we can see that our SHDC protocol is always better than our SDC protocol in both datasets. For the **Gas** dataset, since checking dominance relation in non-leaf nodes only needs to send one ciphertext to $\mathcal{CS}_2$, our SHDC protocol is significantly more efficient in the communication cost when determining dominance relations by non-leaf nodes. For the **Electricity** dataset, although the dominance relation is determined by leaf nodes, and additional communication costs, i.e., the transferred ciphertexts of non-leaf nodes, are incurred, the average transferred ciphertexts of our SHDC protocol are slightly less than that of our SDC protocol. It is because the benefits of determining dominance relations by non-leaf nodes outweigh the incurred costs. This trend will be more obvious when the number of dimensions is large, as shown in the reduced figure in Fig. 14(b).

## VIII. RELATED WORK

Privacy-preserving skyline queries have attracted considerable attention in the database community [8]–[11], [23], [24]. In 2013, Bothe et al. [8] mapped the problem of computing skyline into determining the non-descending series that was computed with the scalar products among sub-vectors of tuples. However, since the simple matrix encryption was adopted to encrypt the sub-vectors of tuples, this scheme is not semantically secure, and an adversary can launch a known plaintext attack to infer the secret keys. Zaman et al. [23] proposed a secure skyline computation scheme in MapReduce. It aimed to support the multi-party secure computation and assumed a trusted party: Coordinator, who can obtain the order of data on each dimension. Since the Coordinator must know the order relations of each dimension to compute skyline, the skyline computation approach in [23] cannot be applied to our scheme due to the security considerations. In 2017, Liu et al. [9] proposed a fully secure skyline computation scheme for dynamic skyline queries. This scheme can protect the plaintexts, single-dimensional privacy, and access patterns from leaking. However, it suffers from the performance issues due to the inefficient basic protocols. Recently, Wang et al. [11] also designed a secure scheme to deal with dynamic

skyline queries. This scheme adopted the order-revealing encryption (ORE) as the cryptographic primitive, leading to the information leakage in single-dimensional privacy. The work in [24] proposed a secure skyline computation scheme for user-defined skyline queries. It also has the problem in performance due to the expensive encryption. Besides, it extended a $d$-dimensional data record to $2d$ dimensions for skyline computation, which is hard to be applied to high-dimensional data. In 2019, Zheng et al. [10] designed a skyline computation protocol by determining dominance relations over encrypted data. However, their work focused on secure data merging technique and did not consider the single-dimensional privacy and access patterns.

## IX. Conclusion

In this paper, we have proposed an efficient and privacy-preserving interval skyline query scheme over encrypted time series data, which can preserve the privacy of plaintexts, single-dimensional privacy, and access patterns while ensuring efficiency. Specifically, we first specified the interval skyline query and introduced the symmetric homomorphic encryption (SHE). Then, we presented SBT and SEQ subprotocols and privacy-preserving logic gates. Based on these building blocks, we designed a secure sort (SS) protocol and a secure dominance check (SDC) protocol. Further, to deal with high-dimensional time series data, we designed a DC-tree and proposed our SHDC protocol. With these protocols, we proposed our secure skyline computation protocol. Finally, we analyzed the security of our scheme and conducted extensive experiments to evaluate it. The results show that our scheme is efficient in both computation and communication.

## References


[1] P. Wang, H. Wang, and W. Wang, "Finding semantics in time series," in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, 2011, pp. 385–396.

[2] Y. Sakurai, Y. Matsubara, and C. Faloutsos, "Mining and forecasting of big time-series data," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 2015, pp. 919–922.

[3] L. Zhang, N. Alghamdi, M. Y. Eltabakh, and E. A. Rundensteiner, "Tardis: Distributed indexing framework for big time series data," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 1202–1213.

[4] N. Alghamdi, L. Zhang, H. Zhang, E. A. Rundensteiner, and M. Y. Eltabakh, "Chainlink: indexing big time series data for long subsequence matching," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 529–540.

[5] B. Jiang and J. Pei, "Online interval skyline queries on time series," in *ICDE*. IEEE, 2009, pp. 1036–1047.

[6] H. Wang, C.-K. Wang, Y.-J. Xu, and Y.-C. Ning, "Dominant skyline query processing over multiple time series," *Journal of Computer Science and Technology*, vol. 28, no. 4, pp. 625–635, 2013.

[7] G. He, L. Chen, C. Zeng, Q. Zheng, and G. Zhou, "Probabilistic skyline queries on uncertain time series," *Neurocomputing*, vol. 191, pp. 224–237, 2016.

[8] S. Bothe, P. Karras, and A. Vlachou, "eskyline: Processing skyline queries over encrypted data," *Proceedings of the VLDB Endowment*, vol. 6, no. 12, pp. 1338–1341, 2013.

[9] J. Liu, J. Yang, L. Xiong, and J. Pei, "Secure skyline queries on cloud platform," in *2017 IEEE 33rd international conference on data engineering (ICDE)*. IEEE, 2017, pp. 633–644.

[10] Y. Zheng, R. Lu, B. Li, J. Shao, H. Yang, and K.-K. R. Choo, "Efficient privacy-preserving data merging and skyline computation over multi-source encrypted data," *Information Sciences*, vol. 498, pp. 91–105, 2019.

[11] W. Wang, H. Li, Y. Peng, S. S. Bhowmick, P. Chen, X. Chen, and J. Cui, "Scale: An efficient framework for secure dynamic skyline query processing in the cloud," in *International Conference on Database Systems for Advanced Applications*. Springer, 2020, pp. 288–305.

[12] M. Naveed, S. Kamara, and C. V. Wright, "Inference attacks on property-preserving encrypted databases," in *SIGSAC*, 2015, pp. 644–655.

[13] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: ramification, attack and mitigation." in *NDSS*. Citeseer, 2012.

[14] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting," in *ICDE*, vol. 3, 2003, pp. 717–719.

[15] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data." in *NDSS*, 2015, p. 4325.

[16] H. Mahdikhani, R. Lu, Y. Zheng, J. Shao, and A. A. Ghorbani, "Achieving o ($\log^3$n) communication-efficient privacy-preserving range query in fog-based iot," *IEEE Internet of Things Journal*, pp. 5220–5232, 2020.

[17] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, "Efficient and privacy-preserving similarity range query over encrypted time series data," *IEEE Transactions on Dependable and Secure Computing*, 2021.

[18] Y. Guan, R. Lu, Y. Zheng, S. Zhang, J. Shao, and G. Wei, "Toward privacy-preserving cybertwin-based spatio-temporal keyword query for its in 6g era," *IEEE Internet of Things Journal*, 2021.

[19] O. Goldreich, *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[20] S. Kamara, P. Mohassel, and M. Raykova, "Outsourcing multi-party computation." *IACR Cryptol. Eprint Arch.*, vol. 2011, p. 272, 2011.

[21] "Greenhouse gas observing network data set," *https://archive.ics.uci.edu/ml/datasets/Greenhouse+Gas+Observing+Network*, 2015.

[22] "Electricity load diagrams data set," *https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014*, 2015.

[23] A. Zaman, M. A. Siddique, and Y. Morimoto, "Secure computation of skyline query in mapreduce," in *International Conference on Advanced Data Mining and Applications*. Springer, 2016, pp. 345–360.

[24] X. Liu, K.-K. R. Choo, R. H. Deng, Y. Yang, and Y. Zhang, "Pusc: privacy-preserving user-centric skyline computation over multiple encrypted domains," in *TrustCom/BigDataSE*. IEEE, 2018, pp. 958–963.